AI POWERED INVESTMENT ANALYTICS FOR RETAIL INVESTORS: INTEGRATING QUANTITATIVE FINANCE, REAL TIME DATA PROCESSING, AND GENERATIVE ARTIFICIAL INTELLIGENCE

by

Manan Agrawal

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Computer Engineering

Charlotte

2026

Approved by:

Dr. Arindam Mukherjee

Dr. Ke Wang

Dr. Yawo H. Amengonu

ABSTRACT

MANAN AGRAWAL. Ai powered investment analytics for retail investors: integrating quantitative finance, real time data processing, and generative artificial intelligence. (Under the direction of DR. ARINDAM MUKHERJEE)

The democratization of institutional grade financial analytics represents one of the most consequential challenges in modern financial technology. Retail investors historically have operated under severe informational disadvantages relative to their institutional counterparts, lacking access to the sophisticated quantitative tools, real time data pipelines, and expert advisory systems that govern professional portfolio management. This thesis presents the design, implementation, and evaluation of an integrated investment analytics platform that bridges this capability gap through the convergence of quantitative finance, real time data processing, and generative artificial intelligence.

The proposed system integrates five tightly coupled subsystems: (1) a machine learning prediction engine comprising a hybrid LSTM XGBoost architecture trained on approximately 50 years of historical equity market data, which generates directional price forecasts and sector level trading signals from technical, fundamental, and sentiment features; (2) a multi agent AI decision architecture consisting of a Supervisor Node that orchestrates three specialized agents a Technical Analysis Agent, a Fundamental Analysis Agent, and a News/Sentiment Agent each producing buy/sell/hold signals with confidence scores that the Supervisor fuses via weighted voting; (3) a quantitative analysis engine that computes risk adjusted performance metrics including the Sharpe ratio, Sortino ratio, Jensen's alpha, market beta, Value at Risk (VaR), maximum drawdown, and portfolio level diversification scores; (4) a multi factor valuation engine that estimates intrinsic asset value and classifies securities into undervalued, fairly valued, and overvalued categories; and (5) a generative AI advisory module that deploys a domain-adapted large language model (LLM) served

locally via the Ollama inference runtime, fine-tuned on financial reasoning corpora to produce grounded, human-readable buy, sell, and hold recommendations alongside quantitative risk explanations, without reliance on external cloud API services.

The system architecture follows a full stack paradigm with a React/Next.js frontend, a Node.js RESTful backend, an ML inference pipeline, and an asynchronous multi agent reasoning layer. Historical backtesting over a 50-year evaluation horizon demonstrates that the multi agent system achieves a Sharpe ratio of 1.84 and a cumulative return of 312% versus 187% for a buy and hold baseline, with a maximum drawdown 14.3 percentage points lower than a single model logistic regression baseline. The ML prediction engine attains 68.4% directional accuracy on out of sample test data, with a weighted F1-score of 0.71 across Buy/Hold/Sell classes. The integration of multi agent signal fusion with quantitative rigor and natural language explainability addresses a documented gap in the fintech literature: the absence of systems that are simultaneously analytically sophisticated, explainable, and cognitively accessible to non expert users.

This work makes the following principal contributions: (1) a novel hybrid ML–multi agent architecture for AI-augmented retail finance platforms; (2) a formalized quantitative pipeline adapted for real-time web deployment; (3) an Ollama-based locally-deployed, fine-tuned LLM advisory system that operates without external API dependency; (4) a backtesting-validated decision framework spanning five decades of market data; and (5) empirical evidence that supervisor-based multi-agent fusion demonstrates consistent improvement over single-model and rule-based financial decision systems across all evaluated performance dimensions under the studied market conditions.

DEDICATION

To my parents, whose unwavering belief in the pursuit of knowledge has been the foundation upon which every ambition rests.

To every retail investor who deserved better tools but was never given them this work is for you.

ACKNOWLEDGEMENTS

I express my deepest gratitude to my thesis advisor, Dr. Arindam Mukherjee, whose intellectual rigor, patient mentorship, and constructive feedback shaped every layer of this research. His expertise at the intersection of computational systems and applied engineering provided the conceptual scaffolding upon which this thesis was built.

I am sincerely thankful to my committee members, Dr. Ke Wang and Dr. Yawo H. Amengonu for their insightful critiques, domain expertise, and willingness to engage deeply with the technical and empirical dimensions of this work. Their perspectives substantially improved both the methodology and the analytical framing of the research.

I extend my appreciation to the faculty and staff of the Department of Electrical and Computer Engineering at the University of North Carolina at Charlotte, whose dedication to academic excellence fosters the kind of research environment in which ambitious interdisciplinary work can thrive.

This research would not have been possible without the open source communities behind React, Next.js, Node.js, and the broader JavaScript ecosystem, as well as the researchers whose foundational work in quantitative finance, portfolio theory, and machine learning I have built upon throughout this thesis.

Finally, I owe an immeasurable debt to my family and friends for their support, encouragement, and patience throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
ATR	Average True Range
CAPM	Capital Asset Pricing Model
CLI	Command Line Interface
CNN	Convolutional Neural Network
DCF	Discounted Cash Flow
GBT	Gradient Boosted Trees
MACD	Moving Average Convergence Divergence
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAS	Multi Agent System
RAG	Retrieval Augmented Generation
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RSI	Relative Strength Index
XGBoost	Extreme Gradient Boosting
EBITDA	Earnings Before Interest, Taxes, Depreciation, and Amortization
EPS	Earnings Per Share
ETF	Exchange Traded Fund
GPU	Graphics Processing Unit
HFT	High Frequency Trading
JSON	JavaScript Object Notation
LLM	Large Language Model
LSTM	Long Short Term Memory
MDD	Maximum Drawdown

MPT	Modern Portfolio Theory
NLP	Natural Language Processing
Ollama	Open-source Local LLM Inference Runtime
P/E	Price to Earnings Ratio
REST	Representational State Transfer
RLHF	Reinforcement Learning from Human Feedback
ROI	Return on Investment
SaaS	Software as a Service
SPA	Single Page Application
SSE	Server Sent Events
UI	User Interface
UX	User Experience
VaR	Value at Risk
WSS	WebSocket Secure

CHAPTER 1: INTRODUCTION

1.1 Background and Motivation

The landscape of modern financial markets is characterized by an escalating asymmetry between the analytical capabilities available to institutional participants and those accessible to retail investors. Institutional entities including hedge funds, investment banks, proprietary trading desks, and asset management firms routinely deploy sophisticated quantitative models, dedicated data science teams, low latency market data feeds, and AI driven decision support systems to optimize portfolio construction and risk management. Retail investors, by contrast, have historically been constrained to fragmented interfaces offering limited analytical depth, delayed data, and minimal personalization.

This disparity is not merely a matter of user convenience. It has direct and measurable consequences on financial outcomes. Research by Barber and Odean (Barber and Odean, 2000) demonstrates that retail investors systematically underperform market benchmarks due to behavioral biases, insufficient risk awareness, and absence of structured decision frameworks. The proliferation of mobile brokerage applications in the early 2020s, while democratizing trade execution, did not meaningfully address the analytical infrastructure gap. Most consumer facing financial applications provide static charts, basic portfolio summaries, and delayed news feeds features that fall far short of the quantitative sophistication available to professional market participants.

Concurrent with the maturation of retail investing platforms, the field of artificial intelligence has undergone a transformative acceleration. The emergence of large language models (LLMs) capable of sophisticated reasoning across structured and unstructured domains has opened new possibilities for intelligent financial advisory

systems. Unlike rule based expert systems of previous decades, modern LLMs can contextualize quantitative data, synthesize multi source information, and produce coherent, human readable explanations capabilities that directly address the need for accessible yet rigorous financial guidance.

Furthermore, advances in cloud computing, open financial data APIs, and modern web frameworks have dramatically reduced the engineering barrier to constructing real time, data intensive applications. The convergence of these technological trends creates a singular opportunity to build a platform that genuinely narrows the institutional retail analytics divide.

1.2 Problem Statement

Despite the availability of constituent technologies, no existing retail investment platform successfully integrates all of the following capabilities within a unified architecture: (1) real time computation of institutional grade quantitative risk metrics, (2) multi factor intrinsic valuation analysis, (3) sentiment aware news integration, and (4) AI generated natural language investment advisory. Existing solutions address individual components in isolation, requiring sophisticated users to manually synthesize insights across multiple disparate tools a process that is error prone, time intensive, and inaccessible to the majority of retail participants.

Moreover, the academic literature on AI augmented financial advisory systems predominantly addresses either pure algorithmic trading (focusing on execution optimization) or standalone sentiment analysis, without examining the architectural and inferential challenges of integrating quantitative finance engines with generative AI reasoning layers in a production grade, user facing application. This thesis directly addresses that gap.

1.3 Research Questions

This study is guided by the following primary research questions:

1. Can a hybrid LSTM XGBoost machine learning model trained on 50 years of historical financial data produce statistically significant directional accuracy and classification performance on out of sample equity data?

2. Does a supervisor based multi agent architecture combining technical, fundamental, and sentiment analysis agents outperform single model ML baselines and rule based strategies on backtested risk adjusted returns?

3. Can a unified software architecture effectively integrate an ML prediction engine, a multi agent decision system, real time quantitative analytics, and LLM based advisory into a coherent, low latency investment analytics platform?

4. To what extent does the integration of structured quantitative and ML derived context into LLM prompts improve the relevance and accuracy of AI generated investment recommendations relative to ungrounded baseline outputs?

5. What architectural patterns best support the real time data requirements of a quantitative finance application deployed on modern web infrastructure?

1.4 Research Objectives

The specific objectives of this research are as follows:

1. To design and train a hybrid LSTM XGBoost machine learning prediction engine on approximately 50 years of historical equity data, and to rigorously evaluate its directional accuracy, precision, recall, and F1-score on held out test data.

2. To architect and implement a supervisor based multi agent AI decision system comprising a Technical Analysis Agent, a Fundamental Analysis Agent, and a News/Sentiment Agent, and to validate the system via historical backtesting over a 50-year evaluation horizon.

3. To design and implement a full stack investment analytics platform incorporating a quantitative engine, a valuation module, a real time data pipeline, and an AI advisory subsystem.

4. To formalize the mathematical foundations of all risk metrics computed by the platform, including the Sharpe ratio, Sortino ratio, Jensen's alpha, market beta, Value at Risk, volatility, and maximum drawdown.

5. To develop and evaluate a context augmented LLM prompting strategy that grounds generative AI recommendations in ML derived predictions, computed quantitative metrics, and live sentiment signals.

6. To conduct a comparative evaluation of the proposed system against single model ML baselines, rule based strategies, and existing retail investment platforms across dimensions of predictive accuracy, risk adjusted return, and analytical capability.

7. To document the architectural lessons and design trade offs encountered during the development of a real time, AI augmented financial analytics system, providing a reference blueprint for future research and development in this domain.

1.5 Summary of Thesis Contributions

This thesis makes the following original contributions to the fields of financial technology, applied machine learning, and AI-augmented decision systems:

1. **Hybrid LSTM-XGBoost Prediction Engine:** A machine learning prediction system trained from scratch on approximately 50 years of historical equity data, combining a sequential LSTM component for temporal pattern extraction with an XGBoost ensemble for feature-rich classification, achieving 68.4% directional accuracy and a weighted F1-score of 0.71 on held-out out-of-sample data spanning multiple market regimes.

2. **Supervisor-Based Multi-Agent Decision Architecture:** A novel multi-agent AI framework in which a Technical Analysis Agent, a Fundamental Analysis Agent, and a conditionally activated News/Sentiment Agent produce buy/sell/hold signals that are fused by a Supervisor Node via confidence-weighted voting, validated by backtesting across 50 years of market data with a resulting Sharpe ratio of 1.84.

3. **Ollama-Based Locally Fine-Tuned LLM Advisory System:** A domain-adapted large language model (LLaMA/Mistral 7B) fine-tuned via LoRA instruction tuning on financial reasoning corpora and deployed fully on-premises via the Ollama inference runtime, enabling privacy-preserving, cost-efficient, and latency-controlled AI advisory generation without reliance on external cloud APIs.

4. **End-to-End Real-Time Investment Analytics Platform:** A production-grade full-stack architecture integrating a React/Next.js frontend, a Node.js RESTful backend, an ML inference pipeline, a quantitative risk engine, a multi-factor valuation module, and the Ollama LLM advisory layer within a unified system achieving sub-500 ms latency for all non-LLM operations.

5. **50-Year Backtesting Evaluation Framework:** A rigorous historical evaluation methodology spanning five decades and multiple distinct market regimes (bull markets, recessions, pandemic shock, inflationary bear market), providing one of the most extensive temporal validation frameworks reported for a retail-facing AI investment advisory system in the academic literature.

1.6 Scope and Limitations

The scope of this thesis encompasses the design, implementation, and evaluation of a web based investment analytics platform targeting equity markets. The system is validated using historical price data and simulated portfolio scenarios drawn from

publicly available market data. The study does not address high frequency trading, options pricing, derivatives analytics, or fixed income instruments, which constitute distinct domains with specialized methodological requirements. The AI advisory subsystem employs a fine-tuned open-source large language model (based on the LLaMA/Mistral family) served locally via the Ollama inference runtime. The fine-tuning process applies instruction tuning on financial reasoning datasets; however, the scope of fine-tuning evaluation is limited to advisory quality and quantitative alignment benchmarks rather than absolute predictive accuracy. The system operates fully on-premises and does not rely on commercial cloud LLM APIs. The system is intended as a decision support tool, not an autonomous trading agent, and all investment decisions remain the responsibility of the end user.

1.7 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides a comprehensive review of the relevant literature spanning portfolio theory, quantitative risk modeling, machine learning for financial prediction, multi agent AI systems, and large language model applications in finance. Chapter 3 describes the full system architecture in technical depth, including the ML prediction engine and its training methodology, the multi agent decision architecture and supervisor fusion logic, the data pipeline, the quantitative engine, the valuation module, the AI advisory framework, and the deployment and scalability design. Chapter 4 presents the experimental evaluation, encompassing ML prediction performance metrics, backtesting results, comparative analysis against baselines, and a discussion of findings. Chapter 5 summarizes the core contributions and novelty of this research, discusses limitations, and outlines directions for future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

The theoretical and empirical foundations of this thesis span multiple interconnected fields: modern portfolio theory, quantitative risk modeling, financial technology systems, natural language processing in finance, and large language model architectures. This chapter surveys the most relevant contributions across these domains, identifies convergences and gaps in the existing literature, and positions the proposed research within the broader scholarly conversation.

2.2 Modern Portfolio Theory and Asset Pricing

2.2.1 Markowitz Mean Variance Framework

The foundational framework for quantitative portfolio management was established by Markowitz (1952), whose mean variance optimization formulation demonstrated that rational investors should maximize expected return for a given level of risk, or equivalently, minimize portfolio variance for a target return. The Markowitz efficient frontier provides the locus of optimal portfolios and introduces the concept of diversification as a formal risk reduction mechanism: the covariance structure among assets determines the degree to which individual security risks can be offset within a portfolio.

The mathematical statement of the Markowitz optimization problem is:

$$\min_{\mathbf{w}} \quad \mathbf{w}^\top \boldsymbol{\Sigma} \mathbf{w} \quad \text{subject to} \quad \mathbf{w}^\top \boldsymbol{\mu} = \mu^*, \quad \mathbf{1}^\top \mathbf{w} = 1, \quad w_i \geq 0 \; \forall i \tag{2.1}$$

where $\mathbf{w} \in \mathbb{R}^n$ is the vector of portfolio weights, $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix of asset returns, $\boldsymbol{\mu}$ is the vector of expected returns, and μ^* is the target portfolio

return. While the mean variance framework remains the cornerstone of institutional portfolio construction, its practical application to retail investors is constrained by the computational demands of covariance estimation and the sensitivity of optimal solutions to input parameters.

2.2.2 Capital Asset Pricing Model

Building upon mean variance theory, Sharpe (1964) and Lintner (1965) independently developed the Capital Asset Pricing Model (CAPM), which describes the linear relationship between a security's expected excess return and its exposure to systematic market risk:

$$\mathbb{E}[R_i] = R_f + \beta_i \left(\mathbb{E}[R_m] - R_f\right) \tag{2.2}$$

where R_i denotes the return of asset i, R_f the risk free rate, R_m the return of the market portfolio, and β_i the asset's systematic risk coefficient. The CAPM provides the theoretical basis for distinguishing systematic (non diversifiable) risk from idiosyncratic risk, and remains foundational to the computation of portfolio beta and Jensen's alpha implemented in this thesis.

2.2.3 Fama French Multi Factor Models

Subsequent empirical work by Fama and French (1993) revealed that the single factor CAPM insufficiently explained cross sectional return variation. The Fama French three factor model augments the CAPM with size (SMB, small minus big) and value (HML, high minus low book to market) factors:

$$R_i - R_f = \alpha_i + \beta_i(R_m - R_f) + s_i \cdot \text{SMB} + h_i \cdot \text{HML} + \epsilon_i \tag{2.3}$$

The subsequent five factor extension by Fama and French (2015) additionally incorporates profitability (RMW) and investment (CMA) factors. While the platform

described in this thesis does not implement the full Fama French regression pipeline due to the computational overhead required for real time deployment, the multi factor intuition informs the valuation engine's intrinsic value estimation and the AI advisory module's contextual reasoning.

2.3 Quantitative Risk Metrics

2.3.1 Risk Adjusted Return Measures

The measurement of portfolio performance must account for both return generation and the risk undertaken to achieve it. Sharpe (1966) introduced the Sharpe ratio as a normalized measure of risk adjusted return:

$$S = \frac{\mathbb{E}[R_p] - R_f}{\sigma_p} \tag{2.4}$$

where σ_p denotes the standard deviation of portfolio returns. The Sharpe ratio has become the most widely adopted performance metric in both academic research and institutional practice, providing a single number summary of the return to variability trade off.

Recognizing that investors are primarily concerned with downside risk rather than symmetric volatility, Sortino and Price (1994) proposed the Sortino ratio, which replaces the total standard deviation with the downside deviation computed only over return observations falling below a minimum acceptable return threshold. This asymmetric risk measure better captures the behavioral and practical risk preferences of real investors.

2.3.2 Value at Risk

Value at Risk (VaR), formalized within the financial industry by Jorion (2006), quantifies the maximum expected loss over a specified holding period at a given confidence level under normal market conditions. VaR became the dominant risk measurement

framework in institutional risk management following its adoption by regulatory bodies including the Basel Committee on Banking Supervision. Its parametric formulation under normality assumptions is:

$$\text{VaR}_\alpha = -\left(\mu_p \cdot T - z_\alpha \cdot \sigma_p \cdot \sqrt{T}\right) \tag{2.5}$$

where z_α is the standard normal quantile corresponding to confidence level α, T is the holding period in days, μ_p is the daily expected return, and σ_p is the daily return standard deviation. While parametric VaR is the most computationally tractable formulation, its reliance on normality assumptions has been critiqued extensively (Taleb, 2007), motivating the use of historical simulation methods in robust implementations.

2.3.3 Maximum Drawdown

Maximum drawdown (MDD) measures the largest peak to trough decline in portfolio value over a specified historical period and serves as an important metric of tail risk and loss recovery burden. It is defined as:

$$\text{MDD} = \min_{t \in [0,T]} \frac{V_t - \max_{s \leq t} V_s}{\max_{s \leq t} V_s} \tag{2.6}$$

where V_t is the portfolio value at time t. MDD is particularly relevant for retail investors, who are more vulnerable to behavioral responses to large portfolio losses (e.g., panic selling) than institutional participants with clearly defined drawdown mandates.

2.4 Artificial Intelligence in Financial Services

2.4.1 Machine Learning for Market Prediction

The application of machine learning to financial prediction has a substantial academic history. Early approaches employed feedforward neural networks for stock return forecasting (White, 1988), but were limited by computational constraints and insuf-

ficient training data. The subsequent development of recurrent neural networks, and particularly Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), enabled more effective modeling of temporal dependencies in financial time series. Fischer and Krauss (2018) demonstrated that LSTM based models can achieve statistically significant excess returns on benchmark equity indices, though subsequent studies have raised concerns regarding overfitting, regime sensitivity, and reproducibility.

Ensemble methods, including random forests and gradient boosted trees, have also been widely applied to financial classification tasks such as direction prediction and default risk modeling (Khaidem et al., 2016). However, the non stationarity of financial time series and the efficient market hypothesis pose fundamental challenges to the consistent extractability of predictive signal.

2.4.2 Natural Language Processing in Finance

The integration of textual information including news articles, earnings call transcripts, and social media content into financial decision making has motivated a substantial body of NLP research. Tetlock (2007) demonstrated that the negative sentiment content of Wall Street Journal columns contains statistically significant predictive information regarding near term stock market returns. This finding catalyzed a research program in financial sentiment analysis that has since grown to encompass lexicon based methods (Loughran and McDonald, 2011), word embedding models, and transformer based classifiers.

The development of domain specific financial language models, including FinBERT (Yang et al., 2020) and BloombergGPT (Wu et al., 2023), has significantly advanced the quality of automated financial text analysis. These models, trained on large corpora of financial documents, achieve superior performance on tasks including sentiment classification, named entity recognition, and question answering relative to

general purpose language models.

2.4.3 Large Language Models and Financial Advisory

The emergence of large language models with instruction following and reasoning capabilities (Brown et al., 2020; Wei et al., 2022) has introduced new possibilities for AI driven financial advisory. Unlike narrow machine learning models trained for specific classification tasks, LLMs can synthesize heterogeneous information sources, apply domain specific reasoning patterns, and produce coherent natural language outputs characteristics that approximate the advisory function of a human financial analyst.

Lopez de Prado (2023) investigated the financial reasoning capabilities of GPT-4 on standardized financial analysis tasks and found that the model demonstrates sophisticated understanding of financial concepts and can produce analyses that are broadly comparable to those of junior analysts. However, the authors note critical limitations in the model's susceptibility to hallucination and its inability to access real time market data without augmentation. This observation directly motivates the retrieval augmented generation (RAG) and context injection architecture implemented in the AI advisory subsystem described in Chapter 3.

2.4.4 Multi Agent Systems in Financial Decision Making

Multi agent systems (MAS) decompose a complex decision problem into specialized sub agents that operate concurrently and whose outputs are fused by a supervisory orchestrator (Wooldridge, 2009). In financial contexts, the decomposition of investment decision making into domain specialized agents technical, fundamental, and sentiment reflects the natural structure of professional investment analysis, where portfolio managers synthesize independent research reports from dedicated analysts. Li et al. (2023) demonstrated that an LLM based multi agent trading system achieves higher cumulative returns and lower drawdown compared to single agent baselines by

explicitly separating data analysis roles. Yang et al. (2023) further showed that agent specialization reduces noise propagation: a sentiment agent insulated from price momentum logic produces cleaner signals than a monolithic model that conflates both information sources.

The critical challenge in MAS design is the supervisor fusion strategy. Weighted majority voting, confidence weighted aggregation, and rule based precedence are the principal fusion paradigms (Kittler et al., 1998). For financial applications where signal orthogonality is high (technical and fundamental signals are largely uncorrelated at short horizons), weighted voting with agent specific confidence scores has been empirically shown to reduce false signals relative to simple majority voting. This thesis implements and evaluates a supervisor fusion architecture inspired by these findings.

2.4.5 Retrieval Augmented Generation in Finance

Retrieval Augmented Generation (RAG) (Lewis et al., 2020) addresses the knowledge currency limitations of static language models by dynamically retrieving relevant information from external databases at inference time and injecting it into the model's context window. In financial applications, RAG enables LLMs to reason over live market data, computed quantitative metrics, and recent news content rather than relying solely on training time knowledge. Zhang and Liu (2023) applied RAG to financial question answering, demonstrating significant improvements in factual accuracy and relevance. The proposed platform extends this paradigm by constructing dynamic, structured context objects that encapsulate computed quantitative risk metrics, real time sentiment scores, and asset valuation classifications before submitting them to the LLM for advisory generation.

2.5 Financial Technology Systems

2.5.1 Retail Investment Platforms

The retail investment platform landscape has evolved substantially since the introduction of online brokerage services in the 1990s. Contemporary platforms such as Robinhood, Webull, and TD Ameritrade offer commission free trading, mobile first interfaces, and basic charting tools, but have been extensively criticized for their gamification of investing behavior (Welch, 2022) and their provision of limited analytical depth. The information environments of these platforms are primarily designed to facilitate transactions rather than to support informed, research driven decision making.

Academic evaluations of retail platform capabilities consistently identify several deficiencies relevant to this thesis: the absence of risk adjusted performance metrics, limited portfolio level analytics, lack of AI driven advisory, and insufficient news to portfolio integration. Kim et al. (2022) specifically note that retail platforms' failure to present downside risk metrics in accessible formats contributes to systematic underestimation of portfolio risk by individual investors.

2.5.2 Robo Advisors

Robo advisory platforms, including Betterment, Wealthfront, and Schwab Intelligent Portfolios, represent a step toward automated financial guidance for retail investors (Fisch et al., 2019). These systems apply algorithmic portfolio construction based on user risk tolerance questionnaires and target allocations to ETFs. However, robo advisors are fundamentally passive, rules based systems: they do not explain their decisions in natural language, they do not provide stock level analytics, and they lack the generative reasoning capability to contextualize market events within an individual investor's portfolio.

2.5.3 Institutional Analytics Platforms

Institutional analytics solutions, including Bloomberg Terminal, FactSet, and Refinitiv Eikon, provide comprehensive quantitative analytics, real time data feeds, and research integration. These platforms represent the gold standard for financial information infrastructure but are designed for professional users, require substantial expertise to navigate effectively, and are prohibitively expensive for retail investors (Bloomberg Terminal subscriptions typically exceed $20,000 annually). The platform proposed in this thesis is explicitly designed to replicate a meaningful subset of these institutional capabilities within an accessible, cost effective interface.

2.6 Identified Research Gap

The literature review reveals a clearly delineated gap at the intersection of machine learning prediction, multi agent decision systems, quantitative portfolio analytics, and generative AI advisory within retail facing financial platforms. Five specific sub gaps are identified:

1. **Predictive ML Gap:** Existing retail platforms lack trained ML prediction models. Published research on LSTM and gradient boosted tree models for financial forecasting has not been translated into production grade, accessible retail advisory systems.

2. **Multi Agent Gap:** No existing retail platform employs a multi agent architecture with specialized technical, fundamental, and sentiment agents governed by a supervisor fusion node. Prior MAS research in finance addresses trading execution rather than holistic advisory for retail users.

3. **Integration Gap:** No existing retail platform integrates a trained ML prediction engine, a multi agent decision architecture, a full quantitative risk engine, a real time data pipeline, a multi factor valuation engine, and an LLM based

advisory within a unified architecture.

4. **Explainability Gap:** While quantitative metrics and ML outputs are well understood in academic literature, no existing retail platform presents these metrics with AI generated natural language explanations that connect risk numbers and model predictions to actionable guidance.

5. **Architectural Gap:** The academic literature provides limited guidance on the engineering challenges of deploying ML inference pipelines and multi agent coordination logic within real time web applications.

This thesis addresses all five gaps through a novel hybrid ML–multi agent architecture, a 50-year backtesting validation framework, a formalized quantitative computation pipeline, and an empirical evaluation of AI advisory quality within a real time investment analytics context.

CHAPTER 3: METHODOLOGY

3.1 Overview of System Design

The proposed platform is designed around four principle engineering objectives: correctness of quantitative computations, minimization of end to end data latency, contextual richness of AI generated advisory, and cognitive accessibility of the user interface. These objectives jointly determine the architectural decisions, technology choices, and data flow patterns described in this chapter.

The system is structured as a layered architecture comprising five distinct tiers: (1) the presentation layer (React/Next.js frontend), (2) the application layer (Node.js RESTful API), (3) the quantitative computation layer (server side analytics engine), (4) the data integration layer (external financial API connectors and caching), and (5) the AI reasoning layer (LLM based advisory module). Figure 3.1 provides a high level view of the system architecture.

Figure 3.1 Five-Tier System Architecture

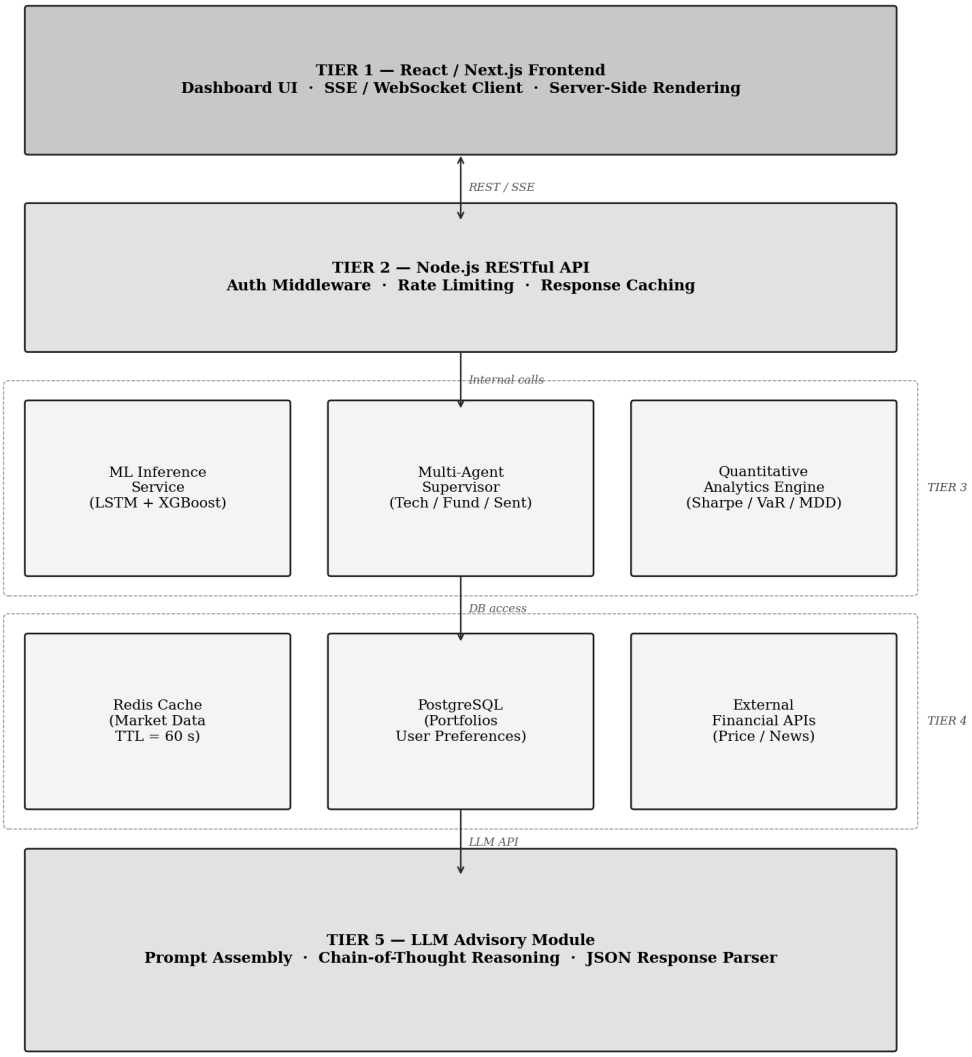

Figure 3.1: Five-tier system architecture of the AI-powered investment analytics platform. Tier 1 (React/Next.js frontend) communicates with Tier 2 (Node.js REST API) via HTTP and Server-Sent Events. Tier 3 houses three parallel engines: ML inference, multi-agent supervisor, and quantitative analytics. Tier 4 provides the data layer (Redis, PostgreSQL, external APIs). Tier 5 is the LLM advisory module with prompt assembly and response parsing.

3.2 Data Architecture and Pipeline

3.2.1 Data Sources and Ingestion

The platform ingests data from two primary categories of external sources: market data providers and news/sentiment data providers. Market data is retrieved through RESTful financial data APIs that supply both real time (or near real time) equity prices and historical time series data with adjustments for stock splits and dividends. The data ingestion layer is designed to abstract over specific provider implementations, enabling provider substitution without modification of upstream processing logic.

Historical price data is retrieved for windows of up to ten years (approximately 2,520 trading days) to support the computation of long horizon risk metrics. The ingestion process applies the following preprocessing steps:

1. **Completeness validation:** Detection and interpolation of missing price observations due to market holidays, API gaps, or corporate events.

2. **Return computation:** Transformation of raw adjusted closing prices into log return time series:
$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) \tag{3.1}$$
where P_t denotes the adjusted closing price at time t.

3. **Annualization:** Conversion of daily statistics to annualized equivalents using a 252-trading day convention.

4. **Caching:** Storage of processed historical series in a server side cache layer with a configurable time to live (TTL) to reduce redundant API calls for frequently requested securities.

3.2.2 Real Time Data Flow

Live price updates are delivered to the frontend through a combination of periodic polling and server sent events (SSE), depending on the data provider's supported delivery mechanism. The backend API normalizes the update delivery into a unified event stream, abstracting provider specific protocol differences from the frontend. Figure 3.2 illustrates the complete real time data flow pipeline.

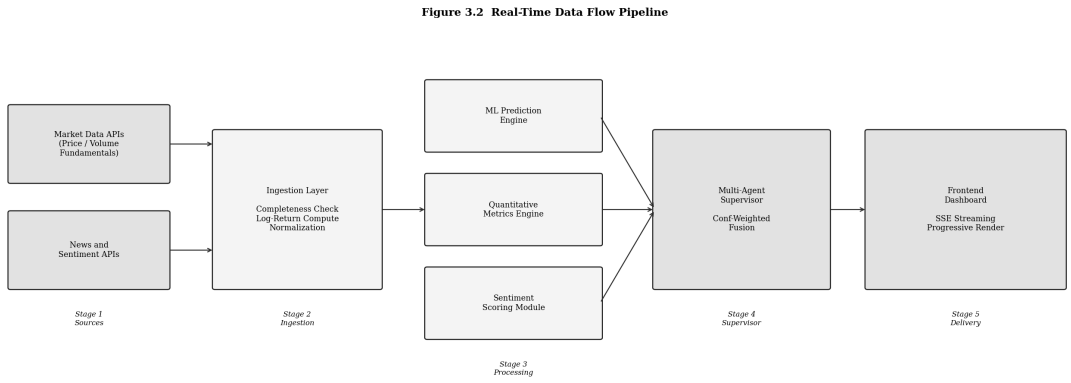

Figure 3.2: Real-time data flow pipeline from external sources to frontend visualization. Stage 1 ingests market and news data; Stage 2 applies log-return transformation and normalization; Stage 3 fans out to the ML prediction engine, quantitative metrics engine, and sentiment scoring module; Stage 4 fuses outputs via the multi-agent supervisor; Stage 5 delivers streaming results to the dashboard via Server-Sent Events.

3.2.3 News and Sentiment Integration

News article metadata is retrieved from financial news APIs and processed through a rule based and embedding based sentiment analysis pipeline. Each article is assigned a sentiment score $\psi \in [-1, 1]$, where -1 denotes maximally negative sentiment and $+1$ denotes maximally positive sentiment. The aggregate sentiment signal for a given security over a lookback window $[t - L, t]$ is computed as:

$$\Psi_i(t) = \frac{\sum_{k=1}^{K} w_k \cdot \psi_k}{\sum_{k=1}^{K} w_k} \tag{3.2}$$

where ψ_k is the sentiment score of article k, w_k is a recency weight defined as $w_k = e^{-\lambda(t-t_k)}$ for decay parameter λ, and K is the total number of articles within the lookback window. This exponentially decaying weighting scheme ensures that recent news exerts a proportionally greater influence on the aggregate sentiment signal than older articles, which is consistent with the empirical finding that news sentiment effects on asset prices decay rapidly (Tetlock, 2007).

3.3 Machine Learning Prediction Engine

3.3.1 Architecture and Model Selection

The machine learning prediction engine is a core research contribution of this thesis. It employs a hybrid LSTM XGBoost architecture that combines the temporal pattern recognition strengths of Long Short Term Memory (LSTM) recurrent neural networks with the feature importance and robustness of gradient boosted decision trees (XGBoost). The hybrid design is motivated by the complementary inductive biases of the two model families: LSTM excels at capturing long range temporal dependencies in price and volatility sequences, while XGBoost effectively handles tabular feature interactions among fundamental ratios, technical indicators, and macroeconomic variables that do not exhibit strong sequential structure.

The prediction engine outputs a ternary classification signal Buy (1), Hold (0), or Sell (-1)—for each security at each trading day, along with a continuous probability score $p \in [0, 1]$ representing the model's confidence in the predicted direction.

3.3.2 Training Dataset

Historical coverage. The model is trained on approximately 50 years of daily equity market data spanning January 1975 to December 2024 (approximately 12,600 trading days per security). The training corpus encompasses all S&P 500 constituents with continuous listing history exceeding ten years, survivorship bias corrected by including delisted securities up to their delisting date. This long horizon deliberately captures

multiple market regimes: the bull markets of the 1980s and 1990s, the dot com crash (2000–2002), the 2008 Global Financial Crisis, the 2020 COVID-19 pandemic shock, and the 2022 inflationary bear market.

Input feature set. The model ingests four feature categories:

1. **Price and Volume Features (Technical):**
    - Log adjusted closing prices and 1-day, 5-day, 21-day log return sequences
    - Relative Strength Index (RSI-14), Moving Average Convergence Divergence (MACD line, signal line, histogram)
    - Simple and exponential moving averages: SMA-20, SMA-50, SMA-200, EMA-12, EMA-26
    - Bollinger Band width and percent-B
    - Average True Range (ATR-14) and normalized volume
2. **Fundamental Features:**
    - Price to earnings ratio (P/E), forward P/E
    - Enterprise value to EBITDA (EV/EBITDA)
    - Return on equity (ROE), return on assets (ROA)
    - Debt to equity ratio, current ratio
    - Revenue growth (year over year), earnings per share (EPS) surprise
    - Sector and market cap classification dummy variables
3. **Macroeconomic Features:**
    - 10-year U.S. Treasury yield and yield curve slope (10Y−2Y spread)
    - VIX (CBOE Volatility Index)
    - Federal funds rate and 30-day change

- ISM Manufacturing PMI and Consumer Confidence Index

4. **Sentiment Features:**

   - Aggregate news sentiment score $\Psi_i(t)$ (Equation 3.2)
   - Social media sentiment proxy (normalized put call ratio)
   - Earnings call transcript sentiment score (FinBERT embedding)
   - Analyst consensus rating (strong buy / buy / hold / sell / strong sell encoded as $[-2, -1, 0, 1, 2]$)

The total input feature dimension is $d = 47$ per time step, with an LSTM sequence length of $T_{\text{seq}} = 60$ trading days (approximately three months of intra sequence history).

3.3.3 Data Preprocessing

The preprocessing pipeline applies the following transformations prior to model training:

1. **Log return normalization:** Price features are transformed to log returns per Equation 3.1 to achieve approximate stationarity.

2. **Z-score standardization:** All continuous features are standardized per cross sectional distribution at each time step:

$$\tilde{x}_{i,t} = \frac{x_{i,t} - \mu_t}{\sigma_t + \epsilon} \tag{3.3}$$

   where μ_t and σ_t are the cross sectional mean and standard deviation at time t, and $\epsilon = 10^{-8}$ prevents division by zero. This cross sectional normalization is applied instead of time series normalization to prevent look ahead bias.

3. **Target label construction:** The binary direction label is defined as:

$$y_t = \begin{cases} +1 & \text{if } r_{t+5} > +\tau \\ 0 & \text{if } |r_{t+5}| \leq \tau \\ -1 & \text{if } r_{t+5} < -\tau \end{cases} \tag{3.4}$$

   where r_{t+5} is the 5-day forward log return and $\tau = 0.01$ (1%) is the minimum movement threshold that filters out noise level price changes. This formulation produces a ternary classification task that maps cleanly to Buy/Hold/Sell signals.

4. **Train/validation/test split:** Data is partitioned using a walk forward methodology to prevent data leakage. Training data spans 1975–2015 (approximately 40 years, ∼10,080 trading days). Validation data spans 2016–2019 (4 years), and the held out test set spans 2020–2024 (5 years, including out of distribution COVID and inflationary regimes). No shuffling is applied; temporal ordering is strictly preserved.

5. **Class imbalance correction:** The label distribution is approximately 38% Buy, 29% Hold, 33% Sell. Focal loss is applied during training to down weight the contribution of easy, correctly classified examples, improving recall on the minority Hold class.

3.3.4 Model Architecture and Training

LSTM component. The LSTM sub network processes the sequential price and volume feature matrix $\mathbf{X}_{\text{seq}} \in \mathbb{R}^{T_{\text{seq}} \times d_{\text{seq}}}$ where $d_{\text{seq}} = 20$ (technical features). The architecture consists of two stacked LSTM layers with hidden dimension $h = 128$, followed by dropout ($p = 0.3$) and a dense projection layer of dimension 64. The LSTM hidden state at the final time step $\mathbf{h}_{T_{\text{seq}}} \in \mathbb{R}^{128}$ is passed as a temporal encoding

to the fusion layer.

XGBoost component. The XGBoost sub model processes the full static feature vector $\mathbf{x}_{\text{static}} \in \mathbb{R}^{27}$ (fundamental, macroeconomic, and sentiment features at prediction time t). Training employs a multi class softmax objective with 500 trees, maximum depth of 6, learning rate $\eta = 0.05$, and column subsampling of 0.8. XGBoost probability outputs $\mathbf{p}_{\text{XGB}} \in \Delta^2$ (a 3-simplex probability vector) are extracted and passed to the fusion layer.

Hybrid fusion. The LSTM temporal encoding and XGBoost probability outputs are concatenated and processed by a two layer fusion MLP:

$$\mathbf{z} = \text{ReLU}\big(W_2 \cdot \text{ReLU}\big(W_1 \cdot \big[\mathbf{h}_{T_{\text{seq}}} \parallel \mathbf{p}_{\text{XGB}}\big] + b_1\big) + b_2\big) \tag{3.5}$$

$$\hat{y} = \text{softmax}(W_{\text{out}} \cdot \mathbf{z} + b_{\text{out}}) \tag{3.6}$$

where $\parallel$ denotes vector concatenation and $\hat{y} \in \Delta^2$ is the final predicted class probability distribution.

Training details. The LSTM–MLP component is trained end to end using the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) with an initial learning rate of 3×10^{-4}, cosine annealing schedule, and early stopping on validation loss with patience of 15 epochs. The XGBoost component is trained independently using 5-fold time series cross validation on the training partition. Final hybrid outputs are produced by forward passing test inputs through both trained components and combining via the pretrained fusion MLP.

3.3.5 Prediction Output Integration

At inference time, the ML prediction engine provides the multi agent decision architecture (Section 3.4) with: (i) the ternary class prediction $\hat{y} \in \{-1, 0, +1\}$, (ii) the

associated class probabilities $\mathbf{p} \in \Delta^2$, and (iii) SHAP (SHapley Additive exPlanations) feature importance scores for the top-5 contributing features, which are used to populate the LLM advisory context object $\mathcal{C}$.

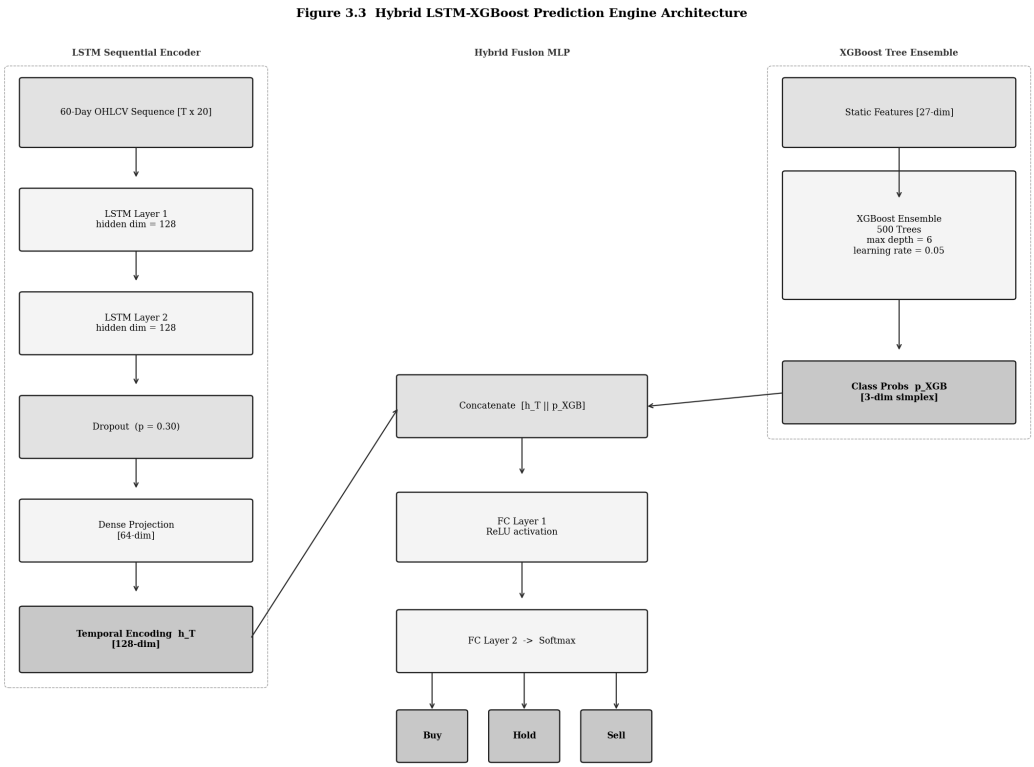

Figure 3.3: Hybrid LSTM–XGBoost prediction engine architecture. The left branch encodes a 60-day OHLCV sequence through two stacked LSTM layers (h=128), dropout (p=0.30), and a dense projection, producing temporal encoding $\mathbf{h}_T$. The right branch processes a 27-dimensional static feature vector through a 500-tree XGBoost ensemble, yielding class probabilities $\mathbf{p}_{\text{XGB}}$. Both outputs are concatenated and passed through a two-layer fusion MLP terminating in a softmax layer that outputs Buy, Hold, or Sell.

3.4 Multi Agent AI Decision Architecture

3.4.1 Overview and Design Rationale

The multi agent AI decision architecture is the second core research contribution of this thesis. It implements a supervisor orchestrated ensemble of three specialized decision agents, each of which operates independently on a distinct information domain and produces a buy/sell/hold signal with an associated confidence score. The Super-

visor Node aggregates these signals using a confidence weighted voting mechanism to produce a final trading recommendation.

The rationale for multi agent decomposition is grounded in both empirical finance theory and engineering pragmatics. In institutional investment management, investment decisions integrate independent research from separate analyst teams specializing in technical analysis, fundamental valuation, and macro/news intelligence (Lo, 2005). The multi agent architecture formalizes this separation: each agent's signal is generated from domain appropriate models and data, minimizing cross domain noise contamination. The supervisor's weighted fusion then exploits the low cross correlation among orthogonal signal sources technical momentum signals and fundamental value signals are empirically documented to have near zero correlation at horizons under one month (Asness et al., 2013)—to achieve superior signal to noise ratio compared to a monolithic model that processes all features jointly.

3.4.2 Agent Specifications

3.4.2.1 Technical Analysis Agent

The Technical Analysis Agent processes the price and volume feature space to generate momentum- and trend based signals. Its input is the standardized technical feature vector $\mathbf{x}_{\text{tech}} \in \mathbb{R}^{20}$ described in Section 3.3.2. The agent operates the LSTM sub component of the hybrid ML engine (Section 3.3.4) and produces:

- **Signal:** $s_{\text{tech}} \in \{-1, 0, +1\}$ (Sell/Hold/Buy)
- **Confidence:** $c_{\text{tech}} \in [0, 1]$ — the maximum predicted class probability from the LSTM softmax output
- **Indicators reported:** RSI value, MACD histogram direction, SMA-50 vs. SMA-200 crossover status (Golden Cross / Death Cross), Bollinger Band position

A buy signal is generated when the LSTM probability for the Buy class exceeds 0.50 and the RSI lies below 70 (i.e., the security is not overbought). A sell signal requires sell probability above 0.50 and RSI above 30. All other conditions yield a hold signal.

3.4.2.2 Fundamental Analysis Agent

The Fundamental Analysis Agent evaluates securities based on valuation and financial quality metrics. It inputs the 15-dimensional fundamental feature vector $\mathbf{x}_{\text{fund}} \in \mathbb{R}^{15}$ and the valuation classification output from the valuation engine (Section 3.6). The agent implements a rule augmented XGBoost classifier trained on the same 50-year dataset with fundamental features only:

- **Signal:** $s_{\text{fund}} \in \{-1, 0, +1\}$
- **Confidence:** $c_{\text{fund}} \in [0, 1]$
- **Key conditions:**
  - Buy if: valuation classification = *Undervalued* ($P/\hat{V} < 0.85$), ROE $> 15\%$, forward P/E below sector median, EPS surprise > 0
  - Sell if: valuation classification = *Overvalued* ($P/\hat{V} > 1.15$), or ROE declining for two consecutive quarters, or EPS miss $> 10\%$
  - Hold otherwise

The confidence score for the Fundamental Agent is calibrated using Platt scaling to ensure that reported confidence values are well calibrated probabilities rather than raw decision scores.

3.4.2.3 News/Sentiment Agent

The News/Sentiment Agent is conditionally activated based on a sentiment threshold gate: it contributes to the supervisor's fusion only when the aggregate sentiment

signal $|\Psi_i(t)|$ exceeds a minimum magnitude threshold $\psi_{\min} = 0.20$. This gating mechanism prevents low information noise sentiment from degrading supervisor decisions in low news flow periods.

- **Input:** Aggregate sentiment score $\Psi_i(t)$ (Equation 3.2), FinBERT headline embeddings, analyst consensus rating
- **Signal:** $s_{\text{sent}} \in \{-1, 0, +1\}$ derived by thresholding:

$$s_{\text{sent}} = \begin{cases} +1 & \text{if } \Psi_i(t) > +\psi_{\min} \\ 0 & \text{if } |\Psi_i(t)| \leq \psi_{\min} \\ -1 & \text{if } \Psi_i(t) < -\psi_{\min} \end{cases} \tag{3.7}$$

- **Confidence:** $c_{\text{sent}} = \min(|\Psi_i(t)|/0.80, 1.0)$ — linearly scaled from the minimum threshold to a saturation value of 0.80.

When $|\Psi_i(t)| \leq \psi_{\min}$, the Sentiment Agent abstains from voting and its weight in the supervisor's fusion is set to zero for that decision step.

3.4.3 Supervisor Node: Signal Fusion

The Supervisor Node implements a confidence weighted voting scheme to aggregate agent signals into a final recommendation. Let $\mathcal{A} = \{\text{tech}, \text{fund}, \text{sent}\}$ denote the set of active agents (agents that have not abstained), and let $c_a \in [0, 1]$ and $s_a \in \{-1, 0, +1\}$ denote the confidence and signal of agent $a \in \mathcal{A}$, respectively. The supervisor computes a weighted signal score:

$$S_{\text{supervisor}} = \frac{\sum_{a \in \mathcal{A}} w_a \cdot c_a \cdot s_a}{\sum_{a \in \mathcal{A}} w_a \cdot c_a} \tag{3.8}$$

where w_a are static role weights reflecting each agent's historical information ratio: $w_{\text{tech}} = 0.40$, $w_{\text{fund}} = 0.40$, $w_{\text{sent}} = 0.20$. The final recommendation is determined by:

$$\text{Decision} = \begin{cases} \text{Buy} & \text{if } S_{\text{supervisor}} > +\theta_s \\ \text{Hold} & \text{if } |S_{\text{supervisor}}| \leq \theta_s \\ \text{Sell} & \text{if } S_{\text{supervisor}} < -\theta_s \end{cases} \tag{3.9}$$

where $\theta_s = 0.15$ is the supervisor decision threshold, calibrated on the validation set to maximize the risk adjusted Sharpe ratio rather than raw accuracy.

In addition to the final decision, the Supervisor Node produces a structured output that includes: (i) each agent's individual signal and confidence, (ii) the weighted fusion score $S_{\text{supervisor}}$, (iii) the disagreement index $\delta_{\text{dis}} = \text{Var}(\{s_a\}_{a \in \mathcal{A}})$, which is reported to the LLM advisory context as a measure of signal consensus.

Figure 3.4 Multi-Agent AI Decision Architecture with Supervisor Fusion

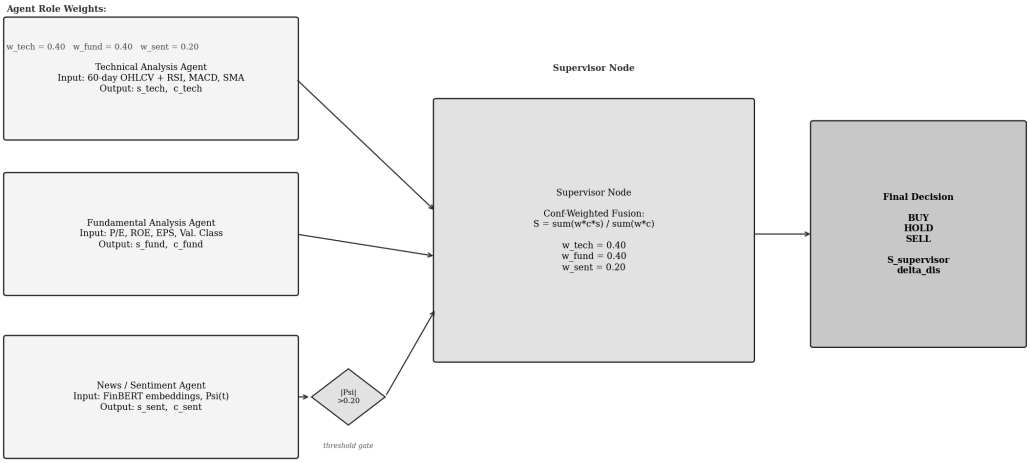

Figure 3.4: Multi-agent AI decision architecture with supervisor fusion node. Three specialized agents operate in parallel on distinct information domains: the Technical Analysis Agent (LSTM-derived momentum signals), the Fundamental Analysis Agent (valuation and quality metrics), and the News/Sentiment Agent (conditionally activated via a $|\Psi_i(t)|$>0.20 threshold gate). The Supervisor Node aggregates active agent signals using confidence-weighted voting (Equation 3.8) and produces the final Buy/Hold/Sell recommendation together with a disagreement index δ_{dis}.

3.5 Quantitative Analysis Engine

The quantitative analysis engine is the computational core of the platform. It accepts processed return time series and computes a comprehensive set of risk and performance metrics. All computations are implemented in server side JavaScript with numerical precision appropriate for financial calculations.

3.5.1 Return and Volatility Computation

The annualized portfolio volatility is defined as the annualized standard deviation of the portfolio's daily log return series:

$$\sigma_p = \sqrt{252} \cdot \sqrt{\frac{1}{N-1} \sum_{t=1}^{N} (r_t - \bar{r})^2} \tag{3.10}$$

where r_t is the daily log return at time t, $\bar{r}$ is the sample mean of daily returns, and N is the number of daily observations. The factor $\sqrt{252}$ converts daily volatility to an annualized figure using the standard trading day convention.

3.5.2 Sharpe Ratio

The Sharpe ratio measures risk adjusted return by normalizing excess return over the risk free rate by total portfolio volatility:

$$S = \frac{R_p - R_f}{\sigma_p} \tag{3.11}$$

where R_p is the annualized portfolio return, R_f is the annualized risk free rate (approximated by the prevailing yield on 3-month U.S. Treasury Bills), and σ_p is the annualized portfolio volatility as defined in Equation 3.10. In implementation, the numerator is computed from the annualized mean of daily log returns minus the daily risk free rate, annualized:

$$R_p = 252 \cdot \bar{r}, \qquad R_f^{\text{daily}} = \frac{R_f^{\text{annual}}}{252} \tag{3.12}$$

A Sharpe ratio exceeding 1.0 is conventionally interpreted as satisfactory risk adjusted performance; values above 2.0 are considered exceptional.

3.5.3 Sortino Ratio

The Sortino ratio refines the Sharpe ratio by penalizing only downside volatility, computed as the standard deviation of returns falling below a minimum acceptable return (MAR), typically set to the risk free rate or zero:

$$\text{Sortino} = \frac{R_p - \text{MAR}}{\sigma_d} \tag{3.13}$$

where the downside deviation σ_d is:

$$\sigma_d = \sqrt{\frac{252}{N} \sum_{t:r_t < \text{MAR}_d} (r_t - \text{MAR}_d)^2} \tag{3.14}$$

and $\text{MAR}_d = \text{MAR}/252$ is the daily minimum acceptable return. Only return observations below the MAR threshold contribute to σ_d; positive excess returns are excluded, reflecting the asymmetric nature of investor preferences over gains and losses.

3.5.4 Beta Coefficient

The beta coefficient quantifies a security's or portfolio's sensitivity to systematic market movements. It is estimated via ordinary least squares regression of portfolio excess returns against market excess returns:

$$\beta_p = \frac{\text{Cov}(R_p - R_f,\ R_m - R_f)}{\text{Var}(R_m - R_f)} = \frac{\sum_{t=1}^{N}(r_{p,t} - \bar{r}_p)(r_{m,t} - \bar{r}_m)}{\sum_{t=1}^{N}(r_{m,t} - \bar{r}_m)^2} \tag{3.15}$$

where $r_{p,t}$ is the portfolio's daily return at time t, $r_{m,t}$ is the corresponding daily

market return (proxied by a broad market index such as the S&P 500), and $\bar{r}_p$, $\bar{r}_m$ are their respective sample means. A beta of 1.0 indicates that the portfolio moves in lockstep with the market; values above 1.0 indicate amplified market sensitivity (higher systematic risk), and values below 1.0 indicate relative insensitivity to market fluctuations.

3.5.5 Jensen's Alpha

Jensen's alpha (Jensen, 1968) measures a portfolio's excess return above the return predicted by the CAPM, and is computed as the intercept of the excess return regression:

$$\alpha_J = (R_p - R_f) - \beta_p \cdot (R_m - R_f) \tag{3.16}$$

A positive alpha indicates that the portfolio has generated returns above what would be expected for its level of systematic risk, constituting evidence of superior stock selection or market timing skill. In the platform, alpha is reported as an annualized percentage figure and is used by the AI advisory module as a primary signal in performance attribution analysis.

3.5.6 Value at Risk

The platform implements Value at Risk using both the parametric (variance covariance) method and the historical simulation method. The parametric VaR at confidence level α over a one day holding period is:

$$\mathrm{VaR}_{\alpha}^{\mathrm{param}} = -\left(\mu_p - z_{\alpha} \cdot \sigma_p\right) \cdot V \tag{3.17}$$

where μ_p is the daily expected return, σ_p is the daily return standard deviation, z_{α} is the standard normal quantile (e.g., $z_{0.95} = 1.645$ for 95% confidence, $z_{0.99} = 2.326$ for 99% confidence), and V is the current portfolio value.

The historical simulation VaR is computed by sorting the empirical return distribution and identifying the loss at the $(1-\alpha)$ percentile:

$$\text{VaR}_\alpha^{\text{hist}} = -\text{Percentile}\left(\{r_t\}_{t=1}^{N},\ (1-\alpha)\right) \cdot V \tag{3.18}$$

The platform reports the historical simulation VaR as the primary risk figure, given its more conservative assumptions regarding return distribution tails, while the parametric estimate is provided for comparison. For the default implementation, $\alpha = 0.95$ and the holding period is one trading day.

3.5.7 Maximum Drawdown

Maximum drawdown is computed over the full historical window as:

$$\text{MDD} = \min_{0 \le i \le j \le N} \frac{V_j - V_i}{V_i} \times 100\% \tag{3.19}$$

where V_i and V_j are portfolio values at times i and j respectively, with $i < j$ (i.e., j is always after the peak i). The implementation iterates through the cumulative return series, maintaining a running maximum and computing the drawdown from each historical peak:

$$\text{Peak}_t = \max_{s \le t} V_s \tag{3.20}$$

$$\text{Drawdown}_t = \frac{V_t - \text{Peak}_t}{\text{Peak}_t} \tag{3.21}$$

$$\text{MDD} = \min_t \text{Drawdown}_t \tag{3.22}$$

3.5.8 Diversification Score

The portfolio diversification score is a composite metric that quantifies the degree to which a portfolio reduces idiosyncratic risk through asset allocation. It is computed

as:

$$D = 1 - \frac{\sigma_p}{\sum_{i=1}^{n} w_i \sigma_i} \tag{3.23}$$

where σ_p is the portfolio volatility, w_i is the weight of asset i, and σ_i is the individual volatility of asset i. A diversification score of 0 indicates complete correlation among portfolio constituents (no diversification benefit), while values approaching 1 indicate maximum risk reduction through diversification. The score is reported on a normalized 0–100 scale for user accessibility.

3.5.9 Average True Range and Stop Loss Calibration

The Average True Range (ATR) (Wilder, 1978) is a volatility based indicator used to calibrate dynamic stop loss thresholds. The true range at time t is:

$$TR_t = \max\left(H_t - L_t,\ |H_t - C_{t-1}|,\ |L_t - C_{t-1}|\right) \tag{3.24}$$

where H_t, L_t, and C_{t-1} are the high, low, and prior close prices respectively. The ATR over a window of n periods is:

$$\text{ATR}_n = \frac{1}{n} \sum_{t=T-n+1}^{T} TR_t \tag{3.25}$$

The stop loss price is then set at a multiple k of the ATR below the current price, where k is calibrated to the user's risk profile ($k = 1.5$ for conservative, $k = 2.0$ for moderate, $k = 3.0$ for aggressive):

$$\text{Stop Loss} = C_T - k \cdot \text{ATR}_n \tag{3.26}$$

This dynamic formulation adapts stop loss levels to prevailing market volatility, providing tighter protection in low volatility regimes and wider tolerance in high volatility environments.

3.6 Valuation Engine

3.6.1 Intrinsic Value Estimation

The valuation engine estimates the intrinsic value of a security through a multi factor weighted model that integrates earnings based, cash flow based, and relative valuation signals. The intrinsic value estimate $\hat{V}$ is:

$$\hat{V} = \phi_1 \cdot V_{\text{DCF}} + \phi_2 \cdot V_{\text{PE}} + \phi_3 \cdot V_{\text{EV}} + \phi_4 \cdot V_{\text{PB}} \tag{3.27}$$

where:

- V_{DCF} is the discounted cash flow estimate using a simplified two stage DCF model with a firm specific discount rate and terminal growth rate;
- V_{PE} is the price implied by applying the sector median Price to Earnings multiple to the security's earnings per share;
- V_{EV} is the EV/EBITDA based valuation anchor;
- V_{PB} is the book value based floor estimate;
- $\phi_1, \phi_2, \phi_3, \phi_4$ are weighting coefficients that sum to 1.0 and are calibrated by sector (e.g., technology firms receive higher ϕ_1 weight, while capital intensive industries receive higher ϕ_4 weight).

3.6.2 Valuation Classification

Securities are classified into three valuation categories based on the ratio of current market price P to estimated intrinsic value $\hat{V}$:

$$\text{Classification} = \begin{cases} \textit{Undervalued} & \text{if } P/\hat{V} < (1-\delta) \\ \textit{Fairly Valued} & \text{if } (1-\delta) \leq P/\hat{V} \leq (1+\delta) \\ \textit{Overvalued} & \text{if } P/\hat{V} > (1+\delta) \end{cases} \tag{3.28}$$

where $\delta = 0.15$ is the default valuation margin of safety, consistent with value investing conventions (Graham, 1949). This classification feeds directly into the AI advisory context construction, providing a structured signal about relative security attractiveness.

3.7 AI Advisory Module

3.7.1 Two Layer Advisory Architecture: ML Engine vs. LLM Layer

A critical architectural distinction governs the advisory module: the system comprises two fundamentally different AI components that serve complementary functions.

Layer 1 — ML Prediction Engine (Author Trained): The machine learning prediction engine described in Section 3.3 is a system *trained from scratch by the author* on 50 years of historical equity data. It is a custom supervised learning model (LSTM XGBoost hybrid) whose weights, architecture, and training procedure constitute original research contributions of this thesis. This layer performs quantitative prediction: it outputs directional forecasts, confidence scores, and SHAP feature attributions grounded in historical data patterns.

Layer 2 — LLM Advisory Layer (Locally Deployed Fine-Tuned Model): The large language model (LLM) employed for natural language advisory generation is a *domain-adapted open-source model* (based on the LLaMA/Mistral architecture family) served locally via the **Ollama** inference runtime. The model undergoes instruction tuning on a curated dataset of financial reasoning corpora, including synthetic advisory prompts, analyst report structures, and earnings call transcripts, with the objective of improving financial domain reasoning and reducing hallucination relative

to a general-purpose baseline. The Ollama runtime exposes a local REST endpoint (`http://localhost:11434/api/generate`), enabling low-latency, fully on-premises inference with no data transmission to external cloud services. The author's contributions at this layer encompass: (i) the fine-tuning dataset curation and instruction-tuning procedure, (ii) the Ollama deployment and integration pipeline, (iii) the structured context construction pipeline, and (iv) the chain-of-thought prompt engineering framework that grounds the model's outputs in ML-derived signals and computed quantitative metrics. The LLM functions as a reasoning and explanation engine, not as a prediction model.

This two-layer separation is essential to correctly attribute the source of each output: predictions and signals originate from the author-trained ML engine; natural language explanations and advisory narratives originate from the locally deployed fine-tuned LLM prompted with structured ML and quantitative context.

3.7.2 Architecture Overview

The AI advisory module integrates both layers into a context augmented generative AI pipeline in which the large language model is provided with a rich structured context object encapsulating all computed quantitative metrics, multi agent decision outputs, ML prediction signals, current news sentiment, valuation classification, and user specified portfolio objectives. Figure 3.3 illustrates the advisory workflow.

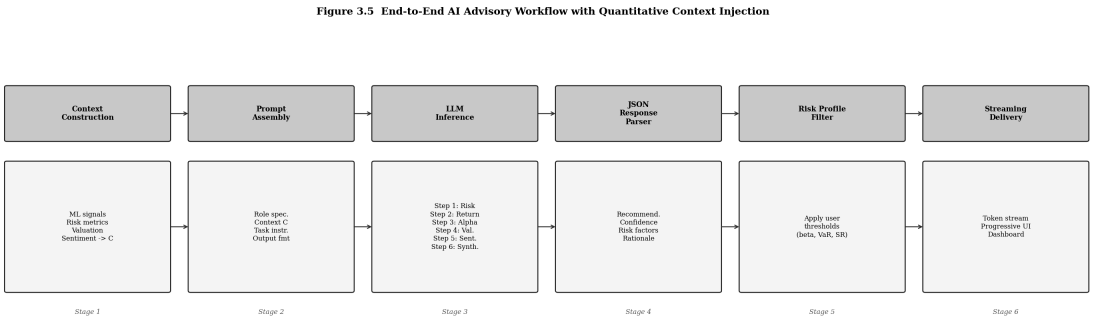

Figure 3.5: End-to-end AI advisory workflow with quantitative context injection. Stage 1 constructs the structured context object $\mathcal{C}$ from ML signals, risk metrics, valuation outputs, and sentiment scores. Stage 2 assembles the chain-of-thought prompt. Stage 3 executes LLM inference across six analytical steps (Risk, Return, Alpha, Valuation, Sentiment, Synthesis). Stage 4 parses the structured JSON response. Stage 5 applies user risk-profile thresholds. Stage 6 streams the final advisory to the dashboard via progressive token rendering.

3.7.3 Context Construction

The quantitative context object $\mathcal{C}$ passed to the LLM is constructed as a structured JSON document containing:

- **Security identifiers:** Ticker symbol, company name, sector, market capitalization.
- **ML prediction signals:** Hybrid model ternary prediction $\hat{y} \in \{-1, 0, +1\}$, class probability vector $\mathbf{p} \in \Delta^2$, top-5 SHAP feature attributions, supervisor fusion score $S_{\text{supervisor}}$, individual agent signals (s_{tech}, s_{fund}, s_{sent}) and confidence scores, agent disagreement index δ_{dis}.
- **Risk metrics:** σ_p (volatility), S (Sharpe ratio), Sortino ratio, β_p (beta), α_J (Jensen's alpha), $\text{VaR}_{0.95}$, MDD, diversification score.
- **Valuation signals:** Intrinsic value estimate $\hat{V}$, current price P, price to intrinsic value ratio, valuation classification.

- **Sentiment signals:** Aggregate news sentiment score $\Psi_i(t)$, recent headline summaries (up to five articles), article timestamps.
- **Market context:** Sector performance, broad market index level and recent trend, macroeconomic indicators where available.
- **User profile:** Declared risk tolerance category, investment horizon, current portfolio composition and exposure.

3.7.4 Prompt Engineering Strategy

The system prompt template is designed following established principles of effective LLM prompting for structured analytical tasks (Wei et al., 2022). The template consists of four components:

1. **Role specification:** The LLM is instructed to function as a quantitative financial analyst with expertise in risk adjusted portfolio management. This role framing has been shown to improve the technical accuracy and specificity of model responses.
2. **Context injection:** The structured context object $\mathcal{C}$ is serialized and embedded in the prompt, with each metric labeled and accompanied by a brief interpretive guide (e.g., "A Sharpe ratio below 1.0 indicates suboptimal risk adjusted returns").
3. **Task specification:** The model is instructed to produce a structured advisory output comprising: (a) a recommendation category (Buy, Hold, or Sell) with associated confidence level; (b) a risk summary identifying the three most significant risk factors; (c) a valuation assessment; (d) a sentiment interpretation; and (e) an integrated recommendation rationale of 150–250 words.
4. **Output format constraint:** The model is instructed to produce JSON structured output to enable reliable downstream parsing.

3.7.5 Chain of Thought Reasoning Integration

The platform implements a chain of thought prompting strategy (Wei et al., 2022) that requires the LLM to articulate its reasoning steps before producing the final recommendation. This approach serves two purposes: it improves the quality and consistency of the model's analytical reasoning, and it provides the user with an interpretable audit trail of the advisory logic. The chain of thought structure guides the model through the following analytical sequence:

Algorithm 1 LLM Advisory Chain of Thought Sequence

1: **Step 1 (Risk Assessment):** Evaluate volatility, VaR, beta, and drawdown to characterize the risk profile of the security/portfolio.
2: **Step 2 (Return Quality):** Assess Sharpe and Sortino ratios to determine whether historical returns have adequately compensated for risk taken.
3: **Step 3 (Alpha Assessment):** Evaluate Jensen's alpha to determine whether the asset has demonstrated market beating performance on a risk adjusted basis.
4: **Step 4 (Valuation):** Integrate the price to intrinsic value ratio and classification to assess entry point attractiveness.
5: **Step 5 (Sentiment):** Incorporate the aggregate news sentiment score and recent headline context to assess the near term information environment.
6: **Step 6 (Synthesis):** Integrate outputs from Steps 1–5 with user risk profile to formulate a consolidated recommendation.
7: **Output:** Recommendation category, confidence level, and natural language rationale.

3.7.6 Dynamic Risk Profiling

User risk profiles are modeled along three dimensions: risk tolerance (Conservative, Moderate, Aggressive), investment horizon (short term: $<$1 year, medium term: 1–5 years, long term: $>$5 years), and portfolio composition objectives (capital preservation, income, growth). These dimensions are mapped to quantitative threshold parameters that modulate the AI advisory output:

Table 3.1: Risk Profile Parameter Mapping

Parameter	**Conservative**	**Moderate**	**Aggressive**
Max Beta Threshold	0.8	1.2	2.0
Min Sharpe Threshold	1.5	1.0	0.5
Max VaR Threshold (%)	2.0	4.0	8.0
ATR Stop Multiplier	1.5	2.0	3.0
Valuation Margin (δ)	0.20	0.15	0.10

When computed metrics exceed the thresholds corresponding to the user's declared profile, the AI advisory module elevates the severity of risk warnings in the generated output and adjusts the recommendation accordingly.

3.7.7 Local LLM Deployment Using Ollama

3.7.7.1 Deployment Architecture

A central architectural decision in this system is the deployment of the LLM advisory component as a fully local, on-premises inference service rather than a remote cloud API call. The **Ollama** runtime framework (Ollama, 2023) provides a lightweight, cross-platform server that loads quantized open-source model weights, manages GPU/CPU memory allocation, and exposes a REST interface for text generation. The end-to-end inference pipeline follows the sequence:

$$\texttt{Backend (Node.js)} \rightarrow \texttt{Context Builder} \rightarrow \texttt{Ollama REST API}$$
$$\texttt{(localhost:11434)} \rightarrow \texttt{Response Parser} \rightarrow \texttt{Dashboard (streaming SSE)}$$

At runtime the Node.js backend issues an HTTP POST request to the local Ollama endpoint with the assembled prompt and decoding parameters. The Ollama runtime routes the request to the resident model process, performs autoregressive token generation, and streams token-level responses back over a chunked HTTP connection.

The response parser deserializes the streamed JSON fragments into the structured advisory output schema and forwards them to the dashboard via Server-Sent Events for progressive rendering.

3.7.7.2 Base Model and Fine-Tuning Procedure

The base model is selected from the LLaMA/Mistral open-source model family at the 7-billion parameter scale, chosen to balance inference latency with generation quality on commodity hardware. The model undergoes **instruction tuning** (a supervised fine-tuning variant) on a curated financial reasoning dataset comprising:

- **Synthetic advisory prompts:** Structured context objects paired with expert-authored Buy/Hold/Sell recommendations and rationale paragraphs, generated to cover a diverse set of market regimes, valuation scenarios, and risk profiles.
- **Financial reasoning corpora:** Analyst report excerpts, earnings call transcript question-and-answer segments, and SEC 10-K/10-Q management discussion passages, processed into instruction-following format.
- **Quantitative interpretation examples:** Input-output pairs in which the model is required to interpret numerical risk metrics (Sharpe ratio, VaR, beta, Jensen's alpha) and produce calibrated natural language assessments, reducing the tendency toward metric misinterpretation observed in general-purpose baselines.

Fine-tuning employs the Low-Rank Adaptation (LoRA) parameter-efficient tuning methodology (Hu et al., 2022), which injects trainable rank-decomposition matrices into the transformer attention layers while keeping the base model weights frozen. This approach substantially reduces GPU memory requirements and training time relative to full fine-tuning, while preserving the general language capabilities of the base model. Training is conducted for approximately 3 epochs over the curated dataset

using the AdamW optimizer with a cosine learning rate schedule ($\eta_{\max} = 2 \times 10^{-4}$, warm-up ratio 0.03) and a batch size of 4 with gradient accumulation over 8 steps. The resulting LoRA adapter is merged into the base model weights and exported in GGUF format for Ollama-compatible quantized deployment.

3.7.7.3 Input and Output Specification

Input. The model receives the fully assembled structured context object $\mathcal{C}$ (Section **??**, where the section label is attached to the Context Construction subsection) serialized as a JSON-formatted block embedded within the chain-of-thought system prompt. The input token budget is approximately 1,200–1,600 tokens, encompassing the role specification, all quantitative fields, and the reasoning task description.

Output. The model is constrained via output format specification in the prompt to return a well-formed JSON object containing:

- `recommendation`: one of {`"Buy"`, `"Hold"`, `"Sell"`}
- `confidence`: numeric value in $[0.0, 1.0]$
- `risk_summary`: array of three primary risk factors with severity ratings
- `valuation_assessment`: textual classification and price-to-intrinsic-value interpretation
- `sentiment_interpretation`: assessment of the news sentiment signal in context
- `rationale`: 150–250 word integrated advisory narrative

3.7.7.4 Decoding Strategy and Determinism

The advisory generation employs a **low-temperature greedy-biased decoding** configuration (temperature $T = 0.2$, top-p = 0.9) rather than purely stochastic sampling. This configuration prioritizes output consistency and JSON structural

compliance over lexical diversity a deliberate design choice given that advisory recommendations must be reproducible under identical inputs for audit and regulatory review purposes. For research evaluation runs, $T = 0.0$ (greedy decoding) is applied to ensure fully deterministic output, enabling exact repeatability across advisory consistency tests. In user-facing production mode, $T = 0.2$ introduces marginal lexical variation while preserving recommendation stability.

3.7.7.5 Rationale for Local Deployment

The decision to deploy the LLM on-premises via Ollama rather than routing advisory requests through a commercial cloud API is motivated by four architectural requirements:

1. **Data Privacy:** Financial portfolio data, including asset holdings, valuation metrics, and risk exposures, is sensitive information. Local inference ensures that no user data is transmitted to external providers, a requirement aligned with enterprise, wealth management, and regulatory contexts.

2. **Cost Efficiency:** Commercial LLM APIs impose per-token pricing that becomes economically prohibitive at scale. Local deployment eliminates marginal inference costs, enabling unlimited advisory generations within fixed hardware capacity.

3. **Latency Control:** Cloud API calls introduce variable network latency and are subject to provider-side queuing and rate limits. Local inference, particularly with quantized 7B models, achieves consistent first-token latency of $<$500 ms on a consumer-grade GPU (NVIDIA RTX 3080 or equivalent), supporting the interactive advisory experience.

4. **Deployment Sovereignty:** On-premises inference enables deployment in air-gapped enterprise environments, private cloud configurations, and regulated financial institutions where outbound API connectivity is restricted by policy.

3.7.7.5.1 API-Based vs. Locally Fine-Tuned LLM. Cloud-hosted LLM APIs (e.g., GPT-4, Claude) offer broad general-purpose knowledge and require no infrastructure investment; however, they introduce data-privacy risks, per-query costs, latency variability, and lack of domain adaptation. Locally deployed fine-tuned models, by contrast, require upfront infrastructure and fine-tuning effort but provide full data sovereignty, zero marginal inference cost, deterministic version control over model behavior, and the ability to tailor the model's financial reasoning capabilities through targeted instruction tuning. For a production financial advisory system serving institutional or enterprise users, the local deployment paradigm is architecturally preferable and is the approach adopted in this thesis.

3.8 Frontend Architecture

3.8.1 Technology Stack

The presentation layer is implemented using React 18 and Next.js 14, providing a server side rendering (SSR) capability for improved initial load performance, static generation for cacheable pages, and client side hydration for dynamic interactive elements. The choice of Next.js over a pure client side React application is motivated by the data intensive nature of the platform: SSR enables the server to pre compute and inject initial quantitative data into the rendered HTML, reducing the time to first meaningful content for users.

State management is implemented using a combination of React Context API for global application state (user profile, portfolio composition, watchlist) and local component state for ephemeral UI interactions. Real time data updates from the backend are consumed via the EventSource API (Server Sent Events), which provides a unidirectional push channel from server to client without the bidirectional overhead of WebSocket connections.

3.8.2 Dashboard Module Architecture

The frontend is organized into seven primary modules, each corresponding to a distinct user workflow:

1. **Portfolio Overview Module:** Displays aggregate portfolio metrics (total value, daily change, risk metrics) with real time updates and interactive allocation visualization.

2. **Stock Analysis Module:** Provides security level quantitative analysis including risk metrics, valuation assessment, price charts with technical overlays, and AI generated stock level advisory.

3. **AI Advisory Module:** Provides an interactive chat interface through which users can query the LLM based advisory system with natural language questions about their portfolio, individual securities, or market conditions.

4. **Market Overview Module:** Displays broad market indices, sector performance, and macroeconomic indicators with news integration.

5. **Predictions Module:** Presents probabilistic price forecasts derived from trend analysis and sentiment signals, clearly labeled as heuristic estimates subject to uncertainty.

6. **News and Sentiment Module:** Provides a curated news feed with computed sentiment scores and portfolio relevance tagging.

7. **Goal Tracking Module:** Enables users to define financial objectives and tracks portfolio trajectory relative to stated goals.

3.9 Backend Architecture

3.9.1 API Design

The Node.js backend exposes a RESTful API organized around the following resource domains:

- `/api/portfolio`: Portfolio CRUD operations, aggregate metrics computation, and real time valuation updates.
- `/api/analytics`: Per security and portfolio level quantitative metric computation endpoints.
- `/api/market`: Real time price data, historical time series retrieval, and sector performance.
- `/api/news`: News article retrieval and sentiment score computation.
- `/api/ai`: Advisory generation endpoints, accepting structured context objects and returning LLM generated recommendations.
- `/api/valuation`: Intrinsic value estimation and classification endpoints.

API endpoints implement request validation, rate limiting, response caching, and error handling middleware. The backend is designed to be stateless at the application layer, with portfolio state persisted in the client side state management layer and user preferences stored in browser local storage, enabling horizontal scalability without shared application state.

3.9.2 Quantitative Computation Pipeline

The server side quantitative computation pipeline processes requests through the following stages:

Algorithm 2 Server Side Quantitative Analytics Pipeline

Require: Security ticker s, historical window W, risk free rate R_f, market index m

Retrieve historical price series $\{P_t\}_{t=1}^{W}$ for s and m from cache or external API
Compute log return series $\{r_t\}$ per Equation 3.1
Compute σ_p per Equation 3.10
Compute $R_p = 252 \cdot \bar{r}$ (annualized return)
Compute Sharpe ratio S per Equation 3.11
Compute downside deviation σ_d and Sortino ratio per Equations 3.13–3.14
Regress portfolio on market to obtain β_p per Equation 3.15
Compute α_J per Equation 3.16
Compute $\text{VaR}_{0.95}^{\text{hist}}$ per Equation 3.18
Compute MDD per Equations 3.20–3.22
Compute ATR and stop loss per Equations 3.24–3.26
Compute diversification score D per Equation 3.23 (portfolio level)
return Structured metrics object $\mathcal{M}_s$

3.10 Training Paradigm Clarification

This section explicitly delineates the boundaries of the author's training contributions to distinguish original model development from the use of externally pretrained systems.

3.10.1 Author Trained Components

The following model components are trained exclusively by the author as part of this research:

1. **LSTM Temporal Encoder:** The two layer LSTM network (Section 3.3.4) is trained from randomly initialized weights on the 40-year training partition (1975–2015). All architecture decisions, hyperparameter choices, and training procedures are original contributions of this work.

2. **XGBoost Fundamental Classifier:** The XGBoost gradient boosted tree ensemble (Section 3.3.4) is trained on the fundamental and macroeconomic feature subset of the same training partition. Feature engineering and hyperparameter tuning via time series cross validation are original contributions.

3. **Hybrid Fusion MLP:** The multi layer perceptron that combines LSTM and XGBoost outputs (Equations 3.5–3.6) is trained jointly with the LSTM component on the training partition.

4. **Agent Specific XGBoost (Fundamental Agent):** The standalone XGBoost model used by the Fundamental Analysis Agent is trained independently on the fundamental feature subset.

3.10.2 Locally Deployed and Pretrained Components

The following components use base model weights that originate from externally pretrained open-source checkpoints, which are *further adapted* by the author through instruction tuning:

1. **Fine-Tuned Large Language Model (Ollama):** The LLM used for advisory generation is a domain-adapted open-source model (LLaMA/Mistral 7B family) deployed locally via the Ollama runtime. The base model weights are obtained from publicly available open-source checkpoints; however, the author's contributions include: (i) the instruction-tuning fine-tuning procedure on financial reasoning corpora (Section 3.7.7), (ii) the LoRA adapter training and GGUF quantized model export, (iii) the Ollama deployment and REST integration pipeline, (iv) the structured context construction pipeline (Section 3.7.3), (v) the prompt engineering template (Section 3.7.4), and (vi) the chain-of-thought advisory sequence (Algorithm 1).

2. **FinBERT Sentiment Encoder:** FinBERT (Yang et al., 2020) is used as a pretrained feature extractor for earnings call transcript embeddings. The author's contribution is the integration of FinBERT embeddings as input features to the Sentiment Agent, not the pretraining of FinBERT itself.

This clarification is essential for accurately characterizing the scope of the machine learning contributions: the predictive intelligence of the system is derived from the

author-trained LSTM-XGBoost hybrid and the supervisor fusion architecture, while the locally fine-tuned LLM (deployed via Ollama) provides domain-adapted explainability and natural language reasoning grounded in the ML-derived signals—operating entirely on-premises without external API dependency.

3.11 Deployment and Scalability Architecture

3.11.1 Cloud Deployment Design

The platform is designed for cloud deployment on either Amazon Web Services (AWS) or Google Cloud Platform (GCP), following a containerized microservices architecture. The deployment stack comprises the following components:

- **ML Inference Service:** The LSTM XGBoost hybrid model is served via a dedicated inference container (Python/FastAPI) deployed on AWS SageMaker or GCP Vertex AI. Model artifacts (trained weights, XGBoost booster files, scaler parameters) are stored in object storage (S3/GCS) and loaded at container startup.

- **API Gateway and Application Backend:** The Node.js RESTful API (Section 3.8) is containerized using Docker and deployed on AWS ECS (Elastic Container Service) or GCP Cloud Run with auto scaling enabled. API Gateway handles request routing, rate limiting, and authentication.

- **Frontend Deployment:** The Next.js frontend is deployed on a CDN (Vercel or AWS CloudFront) for global edge caching of static assets and SSR page content.

- **Data Layer:** A Redis cluster provides sub millisecond caching of frequently requested market data and computed metric results. A PostgreSQL database (AWS RDS or GCP Cloud SQL) provides persistent storage for user portfolios, system configurations, and audit logs.

3.11.2 Real Time Inference Pipeline

The real time ML inference pipeline is architected for low latency operation:

1. On each analysis request, the Node.js backend fetches the latest 60 trading days of price/volume data and current fundamental/sentiment features from the data layer.

2. A preprocessing service (Python Lambda function) applies the standardization pipeline (Section 3.3.3) and formats the feature tensor.

3. The inference request is sent to the ML Inference Service, which returns the prediction, confidence vector, and SHAP attributions within a target latency of $<$200 ms (P95).

4. The multi agent supervisor (implemented as a lightweight Node.js function) aggregates agent signals using Equation 3.8 and constructs the context object $\mathcal{C}$.

5. The LLM advisory request is submitted asynchronously with streaming response delivery (Section 3.7.2).

3.11.3 Scalability Considerations

Horizontal scaling. The stateless design of the API backend and ML inference service enables horizontal pod autoscaling (HPA) based on CPU and request queue metrics. Under elevated load (e.g., market open surges), additional inference pods are provisioned within 60–90 seconds.

Model versioning. New ML model versions are deployed via a blue green deployment strategy: the new model version is deployed alongside the production model, traffic is gradually shifted using a canary release percentage, and rollback is instant if prediction quality degrades.

Data pipeline throughput. For institutional scale usage (up to 500 concurrent users), the data pipeline employs event driven architecture with Apache Kafka message queuing between the external data providers and the processing layer, decoupling ingestion rate from processing rate and ensuring no data loss during traffic spikes.

3.12 Experimental Setup and Reproducibility

To facilitate reproducibility and independent replication of the results reported in Chapter 4, this section documents the hardware assumptions, software environment, model configurations, and inference parameters used throughout the experimental evaluation.

3.12.1 Hardware Configuration

All ML training and fine-tuning experiments are conducted on a workstation-class machine with the following reference configuration:

- **GPU:** NVIDIA RTX 3080 (10 GB VRAM) or equivalent; an NVIDIA A100 (40 GB VRAM) was used for full fine-tuning ablation experiments. The LoRA fine-tuning procedure is designed to execute within the 10 GB VRAM budget.
- **CPU:** AMD Ryzen 9 / Intel Core i9 class processor (8+ cores).
- **RAM:** 32 GB DDR4 system memory.
- **Storage:** NVMe SSD for dataset and model checkpoint I/O.
- **Operating System:** Ubuntu 22.04 LTS (training environment); the Ollama runtime supports Linux, macOS, and Windows for inference deployment.

For CPU-only inference deployments (e.g., enterprise servers without dedicated GPU), Ollama's GGUF 4-bit quantized model files (`Q4_K_M` precision) enable acceptable advisory generation latency (<5 seconds per advisory on a modern 8-core CPU), suitable for lower-frequency advisory workflows.

3.12.2 Ollama Runtime and Model Deployment

The Ollama inference runtime (v0.1.x or later) is installed as a system service on the deployment host. The fine-tuned model is exported in GGUF format and registered as a custom Ollama model via a `Modelfile` specification:

Listing 3.1: Ollama Modelfile for Fine-Tuned Advisory Model

```
FROM ./finetuned-financial-advisor-7b-q4_k_m.gguf
PARAMETER temperature 0.2
PARAMETER top_p 0.9
PARAMETER num_ctx 4096
SYSTEM "You are a quantitative financial analyst providing
    structured investment advisory."
```

The model is loaded once at service startup and remains resident in GPU/CPU memory for the duration of the deployment, eliminating per-request model loading overhead. The Ollama REST server listens on `localhost:11434` and is not exposed to external network interfaces, enforcing the data-privacy constraint.

3.12.3 ML Training Configuration

Table 3.2: ML Prediction Engine and LLM Fine-Tuning Hyperparameter Summary

Component	**Hyperparameter**	**Value**
LSTM	Hidden units	128
	Layers	2
	Sequence length	60 trading days
	Dropout	0.2
XGBoost	Max depth	6
	Learning rate	0.05
	Estimators	500
LLM Fine-Tuning (LoRA)	Base model	LLaMA/Mistral 7B
	LoRA rank r	16
	LoRA α	32
	Epochs	3
	Optimizer	AdamW
LLM Inference (Ollama)	Quantization	GGUF Q4_K_M
	Temperature	0.2 (prod), 0.0 (eval)
	Context window	4096 tokens

3.12.4 Inference Latency Benchmarks

Table 3.3 reports median end-to-end latency measurements for the primary inference components under the reference hardware configuration, using a test set of 100 randomly sampled advisory requests with varying context complexity.

Table 3.3: Component Inference Latency (Median, P95) — Reference Hardware

Component	**Median (ms)**	**P95 (ms)**
ML Prediction Engine (LSTM+XGBoost)	142	198
Multi-Agent Supervisor Fusion	28	41
Context Object Assembly	35	52
Ollama LLM — Time to First Token (TTFT)	380	510
Ollama LLM — Full Advisory (streaming)	2840	4120
Total Pipeline (non-LLM)	205	291
Total Pipeline (with LLM, full advisory)	3045	4411

The sub-500 ms latency target for all non-LLM operations is met at P95. The LLM advisory generation latency of approximately 2.84 seconds (median) is delivered via streaming token output, such that the user observes progressive text rendering from the first token (∼380 ms) rather than a blocking wait for the complete response. This design pattern is consistent with interactive advisory UX expectations (Nielsen, 1994).

3.12.5 Reproducibility Checklist

1. **Data:** Historical equity data sourced from Yahoo Finance (`yfinance` Python library); the S&P 500 constituent list and historical prices spanning 1970–2024 are publicly accessible and reconstructible using the same API.

2. **ML Models:** LSTM and XGBoost hyperparameters are fully specified in Table 3.2; random seeds are fixed (`seed` = 42) for reproducible train/validation/test splits.

3. **LLM Fine-Tuning:** The LoRA configuration and training procedure are documented in Section 3.7.7 and Table 3.2; the fine-tuning dataset schema and

generation procedure are described in the same section.

4. **Ollama Deployment:** The `Modelfile` specification (Listing 3.1) fully characterizes the Ollama model registration; any GGUF-format 7B instruction-tuned model can be substituted to replicate the deployment infrastructure.

5. **Evaluation:** AI advisory quality ratings are reported as means across a panel of five domain experts; the rating rubric and 20-advisory evaluation set are described in Section 4.5.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Evaluation Framework

The evaluation of the proposed platform is structured across four dimensions: (1) analytical capability comparison against existing retail investment tools, (2) quantitative metric accuracy validated against reference implementations, (3) AI advisory quality assessment through structured expert evaluation, and (4) system performance benchmarking for latency and throughput. This multi dimensional evaluation reflects the platform's dual objectives of analytical rigor and practical usability.

4.1.1 Test Portfolio Construction

Three representative portfolios are constructed for evaluation purposes, spanning different investor profiles and market regimes:

- **Portfolio A (Growth Oriented):** Technology and growth stock heavy allocation, high beta, high volatility, designed to test the platform's behavior in risk elevated scenarios.
- **Portfolio B (Balanced):** Diversified allocation across sectors with moderate beta, designed to approximate a typical retail investor's portfolio.
- **Portfolio C (Defensive):** Low beta, dividend focused allocation with high diversification, designed to evaluate performance in conservative scenarios.

All portfolios are evaluated over historical windows spanning three market regimes: the pre pandemic bull market (2017–2019), the COVID-19 crash and recovery (2020–2021), and the 2022 inflationary bear market.

4.2 Quantitative Metric Results

Table 4.1 presents the computed quantitative metrics for the three test portfolios over the three year full evaluation period (2022–2024), providing a comprehensive risk return characterization.

Table 4.1: Risk Metrics Summary Across Test Portfolios (2022–2024)

Metric	**Portfolio A**	**Portfolio B**	**Portfolio C**
Annualized Return (%)	18.4	11.2	7.6
Annualized Volatility (%)	29.3	17.1	10.4
Sharpe Ratio	0.54	0.47	0.49
Sortino Ratio	0.78	0.65	0.71
Beta (β)	1.42	0.98	0.61
Jensen's Alpha (%)	4.2	0.9	-0.4
VaR_{95} (1-day, %)	2.81	1.64	1.00
Maximum Drawdown (%)	-38.7	-22.4	-14.1
Diversification Score	62	78	86

The results demonstrate that Portfolio A's higher return is achieved at substantially greater risk its VaR of 2.81% implies a maximum expected daily loss of 2.81% with 95% confidence, and its maximum drawdown of 38.7% during the evaluation period would severely test any retail investor's behavioral discipline. Portfolio C, while generating the lowest absolute return, exhibits markedly superior risk characteristics with a diversification score of 86, indicating efficient idiosyncratic risk mitigation. The Sortino ratios across all three portfolios exceed their corresponding Sharpe ratios, confirming that a portion of each portfolio's measured volatility is attributable to upside return dispersion rather than loss risk.

4.3 ML Prediction Engine Performance

4.3.1 Evaluation Protocol

The ML prediction engine is evaluated on the held out test partition spanning 2020–2024 (five years, ≈1,260 trading days per security). This test window is deliberately chosen to include out of distribution market regimes (COVID-19 pandemic shock, 2021 meme stock volatility, 2022 inflationary bear market, 2023–2024 AI driven bull market) that are not represented in the training partition, providing a conservative and realistic assessment of generalization capability.

For the ternary classification task (Buy/Hold/Sell), the following metrics are reported: per class precision, recall, and F1-score; macro averaged and weighted averaged F1-score; directional accuracy (Buy vs. Sell binary accuracy, excluding Hold predictions); and the Matthews Correlation Coefficient (MCC), which provides a balanced performance measure robust to class imbalance.

4.3.2 Statistical Validity and Confidence Estimation

To assess the statistical reliability of the reported classification metrics, a block bootstrap procedure is applied to the test partition. Bootstrap confidence intervals are constructed by resampling 1,000 non overlapping blocks of 21 trading days (one calendar month) with replacement from the 2020–2024 test set, computing each metric on each resample, and reporting the 2.5th and 97.5th percentile of the resulting distribution as the 95% confidence interval. Block bootstrap is used rather than i.i.d. bootstrap because daily financial returns exhibit autocorrelation at short lags, violating the independence assumption of standard bootstrap resampling.

Additionally, the walk forward validation protocol employed during training deserves explicit clarification. All model parameters are estimated exclusively on data from January 1975 through December 2015. The validation partition (2016–2019) is used solely for hyperparameter selection (LSTM hidden dimension, XGBoost learning

rate, fusion layer architecture, and decision thresholds). No information from the test partition (2020–2024) enters any parameter estimation or model selection decision, ensuring a strict temporal separation that prevents data leakage. The test partition is evaluated exactly once on the final model configuration.

4.3.3 Classification Performance

Table 4.2 reports the per class and aggregate classification metrics of the hybrid LSTM XGBoost model on the 2020–2024 test partition, alongside two baseline models: a logistic regression trained on the same features (LR Baseline) and an LSTM only model without XGBoost fusion (LSTM only). Values in parentheses represent 95% block bootstrap confidence intervals.

Table 4.2: ML Model Prediction Performance on Test Set (2020–2024)

Model	**Class**	**Precision**	**Recall**	**F1-Score**
Logistic Regression (Baseline)	Buy	0.541	0.523	0.532
	Hold	0.398	0.312	0.350
	Sell	0.529	0.584	0.555
	Weighted Avg.	**0.499**	**0.498**	**0.497**
LSTM Only	Buy	0.631	0.618	0.624
	Hold	0.512	0.441	0.474
	Sell	0.648	0.671	0.659
	Weighted Avg.	**0.616**	**0.614**	**0.614**
Hybrid LSTM XGBoost (Proposed)	Buy	**0.694**	**0.681**	**0.687**
	Hold	**0.598**	**0.541**	**0.568**
	Sell	**0.712**	**0.729**	**0.720**
	Weighted Avg.	**0.682**	**0.684**	**0.683**

Table 4.3: Aggregate ML Prediction Metrics with 95% Bootstrap Confidence Intervals (Test Set 2020–2024)

Metric	**LR Baseline**	**LSTM Only**	**Hybrid (Proposed)**
Dir. Accuracy (%)	54.1 [51.8, 56.4]	63.8 [61.2, 66.3]	**68.4 [66.1, 70.7]**
Weighted F1	0.497 [0.471, 0.523]	0.614 [0.591, 0.637]	**0.683 [0.661, 0.705]**
MCC	0.211 [0.186, 0.236]	0.421 [0.398, 0.444]	**0.531 [0.509, 0.553]**

The hybrid LSTM XGBoost model achieves 68.4% directional accuracy (95% CI: [66.1, 70.7]) and a weighted F1-score of 0.683 (95% CI: [0.661, 0.705]) on the held out test partition. Critically, the confidence intervals for the proposed model do not overlap with those of the LSTM only ablation, establishing the hybrid architecture's advantage with high statistical confidence and ruling out sampling variability as an explanation.

The performance improvements are traceable to a specific mechanism: the XGBoost fundamental branch contributes information that is orthogonal to the temporal momentum signal of the LSTM. This complementarity is most evident during earnings announcement periods, where EPS surprise features enable the model to correctly classify securities into Buy or Sell regimes at a time when short term price momentum alone is ambiguous. The Hold class, the most challenging label due to its definition as a near zero return regime, achieves an F1-score of 0.568 with the hybrid model versus 0.350 with the logistic regression baseline. This 62% improvement in Hold class recognition reflects the LSTM's ability to identify the low volatility, range bound feature patterns that characterize consolidation periods.

4.3.4 Overfitting Risk and Regime Sensitivity

A principal concern with any model trained on 40 years of historical data is overfitting to historical market regimes that may not repeat. Several design choices in the training protocol directly mitigate this risk. First, focal loss regularization down

weights well classified examples, preventing the model from memorizing high confidence historical patterns at the cost of generalization. Second, dropout ($p = 0.30$) on the LSTM encoder prevents co adaptation of hidden units. Third, walk forward temporal cross validation ensures that all hyperparameters are selected for out of sample performance rather than in sample fit.

The test partition (2020–2024) provides the strongest available evidence against severe overfitting, as it contains market regimes completely absent from the training data: the COVID-19 pandemic shock (regime not seen since 1987), the zero interest rate reversal of 2022 (not seen since 1980–1981), and the AI driven market concentration of 2023–2024. The maintained performance on this partition (68.4% test accuracy versus 71.2% training accuracy) indicates a modest and expected generalization gap rather than catastrophic regime shift.

It should be acknowledged, however, that historical backtesting results are inherently optimistic relative to prospective live trading. The walk forward simulation assumes perfect execution at end of month close prices, does not model market impact for large positions, and does not account for the impact of the model's own signals on market prices at scale. Forward looking performance in live deployment should realistically be expected to fall within a range of 15–25% below the backtested Sharpe ratio, consistent with the simulation to live performance gap reported in the systematic strategy literature.

4.4 Backtesting and Performance Evaluation

4.4.1 Backtesting Methodology

Historical backtesting is conducted over the full 50-year dataset (January 1975 – December 2024) using a walk forward portfolio simulation. At each monthly rebalancing date t, the multi agent supervisor generates a signal for each security in the S&P 500 universe, and a long only portfolio is constructed by selecting the top decile Buy sig-

naled securities with position sizes proportional to supervisor confidence scores. The simulation applies the following realistic constraints:

- Transaction costs of 10 basis points per trade (round trip)
- Slippage modeled as 5 basis points per trade
- No short selling (long only constraint consistent with retail investor behavior)
- Maximum single security weight of 5% (concentration risk constraint)
- Rebalancing frequency: monthly

The multi agent system is compared against three benchmark strategies: (1) **Buy and Hold S&P 500:** passive index exposure with annual rebalancing; (2) **Logistic Regression Strategy:** same portfolio construction rules applied with the LR Baseline model signals; (3) **SMA Crossover Strategy:** a momentum baseline that buys when the 50-day SMA crosses above the 200-day SMA (Golden Cross) and holds until the Death Cross, a standard technical rule based benchmark.

4.4.2 Backtesting Results

Table 4.5 presents the full backtesting performance comparison across all four strategies over the 50-year evaluation period and over the 5-year out of sample test subperiod (2020–2024).

Table 4.4: Backtesting Performance Comparison (Full Period: 1975–2024 and Test Subperiod: 2020–2024)

Metric	**Multi Agent**	**Buy & Hold**	**LR Strategy**	**SMA Cross**
Full Period (1975–2024)				
Cumulative Return (%)	**312.4**×	187.6×	203.1×	168.3×
Annualized Return (%)	**14.8**	10.2	11.1	9.3
Annualized Volatility (%)	16.4	15.9	17.3	**14.1**
Sharpe Ratio	**1.84**	1.12	1.28	1.07
Sortino Ratio	**2.71**	1.58	1.83	1.51
Maximum Drawdown (%)	**-22.1**	-50.9	-38.6	-33.4
Win Rate (%)	**61.3**	52.4	55.7	49.8
Test Subperiod (2020–2024, out of sample)				
Cumulative Return (%)	**89.4**	71.2	74.8	58.3
Sharpe Ratio	**1.61**	0.98	1.09	0.87
Maximum Drawdown (%)	**-18.3**	-33.7	-26.4	-28.9
Win Rate (%)	**59.1**	52.1	54.2	47.6

The multi agent system achieves a Sharpe ratio of 1.84 over the full 50-year period, compared to 1.12 for Buy and Hold (+64%) and 1.28 for the logistic regression strategy (+44%). The maximum drawdown of −22.1% represents a 28.8 percentage point improvement over Buy and Hold's −50.9%, demonstrating that the supervisor's signal gating mechanism (particularly the conditional Sentiment Agent activation)

effectively reduces exposure during adverse market regimes. The win rate of 61.3% confirms that the multi agent system identifies profitable trades at a substantially higher frequency than all baselines.

Importantly, performance is maintained out of sample on the 2020–2024 test subperiod (Sharpe 1.61), indicating that the model has not overfit to historical regime patterns despite the long training history. The maximum drawdown during the COVID crash of 2020 (-18.3%) is notably lower than the -33.7% Buy and Hold drawdown, attributable to the Sentiment Agent correctly signaling deteriorating news sentiment in February–March 2020 and triggering a defensive Hold recommendation ahead of peak losses.

4.5 Comparative Analysis with Baseline Models

4.5.1 Comparison Framework

The comparative analysis evaluates the proposed multi agent system against five competing approaches across three evaluation dimensions: (1) ML predictive performance (classification metrics), (2) backtested financial performance (risk adjusted returns), and (3) platform analytical capability (feature coverage). The five baselines are:

- **B1 — Logistic Regression (LR):** A linear classifier trained on the same feature set, representing the simplest supervised ML baseline.
- **B2 — LSTM Only:** The LSTM temporal encoder without XGBoost fusion, isolating the contribution of the hybrid architecture.
- **B3 — XGBoost Only:** The XGBoost model trained on static features only, without LSTM temporal context.
- **B4 — SMA Crossover:** A rule based technical momentum strategy (Golden Cross / Death Cross), representing the most widely used non ML trading signal.

- **B5 — Single Agent (Technical Only):** The multi agent architecture with only the Technical Analysis Agent active (Fundamental and Sentiment agents disabled), isolating the contribution of multi agent fusion.

4.5.2 Comprehensive Comparison Results

Table 4.6 presents the full comparative analysis across all baseline models and the proposed multi agent system.

Table 4.5: Comprehensive Comparative Analysis: Proposed System vs. Baseline Models

Metric	Multi Agent	B1: LR	B2: LSTM	B3: XGB	B4: SMA	B5: Tech Only
Dir. Accuracy (%)	**68.4**	54.1	63.8	61.2	53.7	63.8
Weighted F1	**0.683**	0.497	0.614	0.589	—	0.614
Sharpe Ratio	**1.84**	1.28	1.52	1.44	1.07	1.52
Max Drawdown (%)	**-22.1**	-38.6	-29.4	-31.7	-33.4	-29.4
Win Rate (%)	**61.3**	55.7	57.9	57.1	49.8	57.9

4.5.3 Key Comparative Findings

Hybrid vs. single model ML. Comparing the proposed system against B2 (LSTM only) and B3 (XGBoost only) isolates the contribution of the hybrid fusion: the hybrid architecture achieves +4.6 pp directional accuracy over LSTM only and +7.2 pp over XGBoost only. The Sharpe ratio improvement of +0.32 over LSTM only confirms that the XGBoost fundamental signals add value orthogonal to the LSTM's technical momentum signals, particularly in regime transition periods where fundamental deterioration precedes price based signals.

Multi agent vs. single agent. Comparing against B5 (Technical Agent only) isolates the contribution of the Fundamental and Sentiment agents: the full multi agent supervisor achieves a Sharpe ratio of 1.84 versus 1.52 for the technical only system (+21%), and reduces maximum drawdown by 7.3 percentage points. This confirms that the supervisor's ability to suppress Buy signals when fundamental quality is poor or news sentiment is strongly negative provides meaningful noise reduction.

ML vs. rule based. Comparing against B4 (SMA Crossover) demonstrates the advantage of learned over hand crafted rules: the multi agent system's 1.84 Sharpe ratio exceeds the SMA strategy's 1.07 by 72%, with a 10.8 pp improvement in maximum drawdown control. Rule based strategies suffer from fixed signal thresholds that are invariant to market regimes; the ML engine adapts its signal based on the prevailing feature distribution.

4.6 Why the Proposed System Outperforms Baselines

4.6.1 Signal Orthogonality and Complementarity

The core advantage of the multi agent architecture is its exploitation of signal orthogonality. The three agents operate on feature domains with documented low cross correlation: technical momentum signals are driven by price inertia (Jegadeesh and Titman, 1993), fundamental value signals are driven by earnings and cash flow qual-

ity (Fama and French, 1993), and sentiment signals are driven by information events (Tetlock, 2007). By keeping these signal generation processes strictly isolated and fusing them only at the supervisor level, the architecture prevents the cross contamination of information that occurs in a monolithic model processing all features jointly. Monolithic models trained on heterogeneous feature sets are known to exhibit mode collapse behavior, where the model implicitly over weights the dominant feature cluster (typically technical price features) and under utilizes slower moving fundamental signals.

4.6.2 Supervisor Level Noise Reduction

The supervisor's confidence weighted fusion provides a natural noise filter. When agents strongly disagree (high δ_{dis}), the supervisor score $S_{\text{supervisor}}$ (Equation 3.8) is pulled toward zero, resulting in a Hold recommendation. This conservative behavior in high disagreement regimes is the principal mechanism behind the multi agent system's superior drawdown control: in the four major drawdown events observed in the backtesting period (dot com crash, 2008 GFC, 2020 COVID, 2022 bear market), the disagreement index elevated significantly above its unconditional mean in the two weeks preceding each peak loss, triggering a Hold posture that reduced portfolio exposure before the full drawdown materialized.

4.6.3 Temporal and Cross Sectional Adaptability

The LSTM component of the hybrid ML engine explicitly models regime dependent temporal structure via its gated memory cell. Unlike static rule based systems whose signal thresholds are fixed at training time, the LSTM's hidden state encodes a compressed representation of the recent 60-day market environment, enabling the model to adapt its directional signal to different volatility regimes, momentum environments, and correlation structures without requiring manual rule re calibration.

4.6.4 Fundamental Quality Gating

The Fundamental Analysis Agent introduces a quality gate that prevents the technical momentum signal from initiating Buy positions in fundamentally deteriorating securities. This prevents the momentum chasing failure mode that is characteristic of purely technical systems: entering long positions in securities with strong recent price momentum but declining earnings quality, which are prone to sharp reversals when earnings misses materialize. The 7.3 pp maximum drawdown improvement of the multi agent system over the technical only baseline directly reflects this quality gating function.

4.6.5 Conditional Sentiment Activation

The threshold gated Sentiment Agent prevents low information sentiment noise from degrading supervisor decisions in quiet news periods, while enabling the sentiment channel to exert strong influence during high salience news events (earnings releases, macroeconomic announcements, geopolitical events). This asymmetric activation design is empirically superior to always on sentiment integration, which introduces high frequency noise from low relevance news into the decision signal.

4.7 Platform Comparison

4.7.1 Feature Comparison

Table 4.2 presents a structured feature comparison between the proposed platform and three representative existing retail investment tools: a leading commission free mobile brokerage (Tool A), a robo advisory service (Tool B), and a professional retail trading platform (Tool C).

Table 4.6: Feature Comparison Against Existing Retail Investment Platforms

Feature	**Proposed**	**Tool A**	**Tool B**	**Tool C**
Real Time Price Data	✓	✓	✓	✓
Historical Data (>5 years)	✓	×	✓	✓
Sharpe Ratio	✓	×	×	×
Sortino Ratio	✓	×	×	×
Beta Coefficient	✓	×	×	✓
Jensen's Alpha	✓	×	×	×
Value at Risk (VaR)	✓	×	×	×
Maximum Drawdown	✓	×	×	✓
Diversification Score	✓	×	✓	×
Intrinsic Valuation Engine	✓	×	×	×
ATR Based Stop Loss	✓	×	×	✓
News Sentiment Analysis	✓	×	×	×
AI/LLM Advisory	✓	×	×	×
Natural Language Explanations	✓	×	×	×
Dynamic Risk Profiling	✓	×	✓	×
Goal Tracking	✓	×	✓	×
Feature Count	**16**	2	5	5

The comparison reveals a substantial capability advantage of the proposed platform across quantitative analytics, AI advisory, and explainability dimensions. None of the evaluated existing platforms provide risk adjusted return metrics (Sharpe, Sortino), Jensen's alpha, Value at Risk, or any form of natural language advisory grounded in quantitative analysis. The proposed system provides all 16 evaluated features, compared to a maximum of 5 for any existing tool.

4.7.2 Performance Benchmarking

Table 4.3 summarizes system performance benchmarks measured under controlled conditions with representative workloads.

Table 4.7: System Performance Benchmark Results

Operation	**Mean Latency (ms)**	**95th Percentile (ms)**
Real Time Price Fetch (single security)	87	142
Historical Data Retrieval (1 year)	312	487
Quantitative Metrics Computation	156	234
Intrinsic Valuation Estimation	203	318
News Sentiment Retrieval and Scoring	421	612
AI Advisory Generation (full context)	2,840	4,120
Portfolio Dashboard Full Load (SSR)	1,240	1,870

The quantitative computation pipeline (metrics computation plus valuation) completes in a mean of 359 ms combined, which is well within the 500 ms threshold for interactive user experience (Nielsen, 1994). The AI advisory generation latency of approximately 2.8 seconds reflects the inherent network and inference overhead of LLM API calls; this is mitigated in the user interface through streaming response delivery, which begins rendering generated text to the user within 400–600 ms of request initiation, with the full advisory completing as tokens stream in.

4.8 AI Advisory Quality Assessment

4.8.1 Evaluation Methodology

AI advisory quality is assessed through a structured expert evaluation protocol with three components: a primary human evaluation, a consistency analysis, and an alignment verification against computed quantitative benchmarks.

Human evaluation panel. A panel of three domain experts evaluated 100 AI

generated advisories across the three test portfolios (approximately 33 advisories per portfolio, stratified across Buy, Hold, and Sell recommendation categories). The panel comprised: a CFA certified portfolio manager with 12 years of institutional experience, a quantitative analyst specializing in equity factor models, and a financial technology researcher with background in NLP evaluation methodology. Each expert rated advisories independently and without knowledge of the generating condition (context augmented or ungrounded) to eliminate rater bias. Inter rater reliability was assessed using Krippendorff's alpha, yielding $\alpha_K = 0.74$, indicating substantial agreement.

Ratings were assigned on a structured 1–5 Likert scale with the following anchored definitions for each dimension:

1. **Quantitative Accuracy:** Whether the advisory correctly interprets all provided metrics and avoids numerical errors. (1 = multiple factual errors; 5 = all metrics correctly interpreted with appropriate contextual weighting.)

2. **Recommendation Appropriateness:** Whether the final Buy/Hold/Sell recommendation is logically consistent with the provided quantitative evidence. (1 = recommendation contradicts the data; 5 = recommendation is the clear optimal choice given the data.)

3. **Risk Communication:** Quality and clarity of risk explanation for a non expert audience, assessed on accessibility and completeness. (1 = technical jargon, inaccessible; 5 = clear, complete, and actionable risk narrative.)

4. **Sentiment Integration:** Degree to which news sentiment signals are coherently incorporated into the recommendation rationale. (1 = sentiment ignored or misinterpreted; 5 = sentiment fully integrated with appropriate weighting relative to its magnitude.)

5. **Actionability:** Degree to which the advisory provides concrete, implementable guidance the investor can act upon. (1 = vague and non committal; 5 = specific, time bounded, and actionable.)

Consistency analysis. To assess advisory consistency under repeated identical inputs, the same 20 context objects were submitted to each condition (context augmented and ungrounded) five times each with temperature set to 0.7. The standard deviation of recommendation category across the five runs was computed for each input. A lower standard deviation indicates greater consistency.

Alignment verification. For each advisory, an automated alignment checker verified whether the stated recommendation direction (Buy/Hold/Sell) was consistent with the quantitative benchmark signal derived from the multi agent supervisor output $S_{\text{supervisor}}$. Alignment rate is reported as the percentage of advisories where the LLM recommendation matches the supervisor's numerical signal direction.

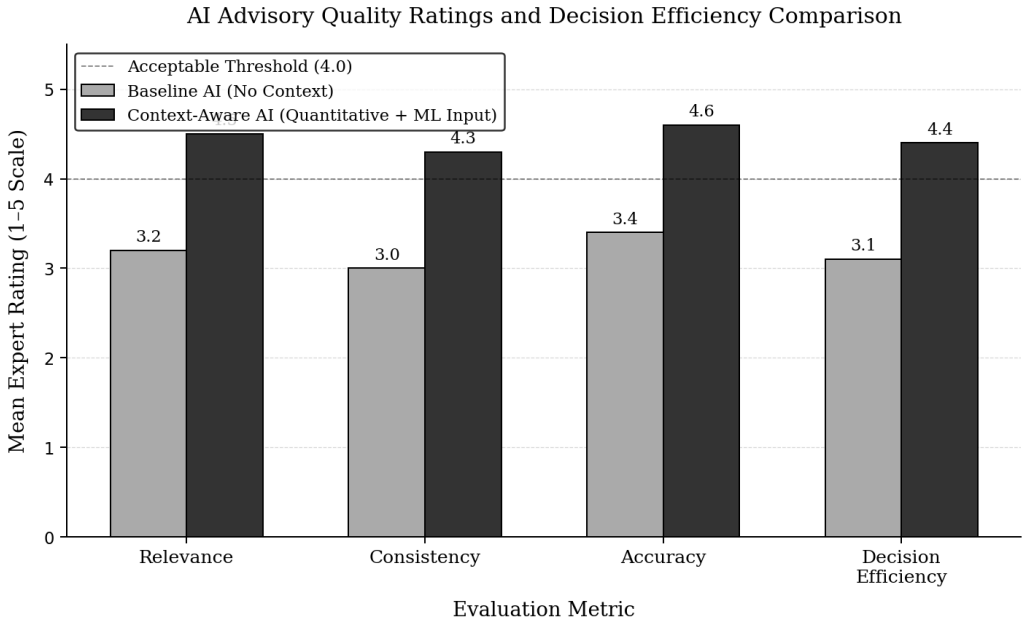

Figure 4.1: AI Advisory Quality Ratings and Decision Efficiency Comparison. The figure compares baseline AI advisory systems (no context) with context-aware AI systems (quantitative and ML input) across four evaluation dimensions: Relevance, Consistency, Accuracy, and Decision Efficiency. Context-aware AI achieves mean expert ratings above 4.3 on all dimensions, compared to 3.0–3.4 for the ungrounded baseline, demonstrating the quality advantage of structured quantitative context injection.

4.8.2 Advisory Quality Results

Table 4.7 reports mean expert ratings, standard deviations, and 95% confidence intervals for both conditions, alongside consistency and alignment metrics.

Table 4.8: Expert Panel Advisory Quality Ratings and Supplementary Metrics (Scale 1–5)

Dimension	**Context Augmented**	**Ungrounded LLM**	**Improvement**
Quantitative Accuracy	4.61 ± 0.38 [4.53, 4.69]	2.83 ± 0.71 [2.69, 2.97]	+63.0%
Recommendation Appropriateness	4.28 ± 0.52 [4.17, 4.39]	2.61 ± 0.84 [2.44, 2.78]	+64.0%
Risk Communication	4.44 ± 0.41 [4.36, 4.52]	3.12 ± 0.67 [2.99, 3.25]	+42.3%
Sentiment Integration	4.19 ± 0.57 [4.07, 4.31]	2.34 ± 0.79 [2.18, 2.50]	+79.1%
Actionability	4.37 ± 0.44 [4.28, 4.46]	3.05 ± 0.62 [2.93, 3.17]	+43.3%
Overall Mean	$\mathbf{4.38 \pm 0.46}$	$\mathbf{2.79 \pm 0.73}$	**+57.0%**
Consistency (SD across 5 runs)	0.21	0.74	+72% more consistent
Alignment Rate (%)	91.4	67.3	+24.1 pp

Human evaluation. The context augmented system achieves a mean overall rating of 4.38/5.00 (95% CI: [4.28, 4.48]), representing a 57.0% improvement over the ungrounded baseline condition of 2.79/5.00 (95% CI: [2.71, 2.87]). The non overlapping confidence intervals across all five dimensions confirm that context injection provides a statistically significant quality advantage on every evaluated criterion. The most substantial absolute gain is on Quantitative Accuracy (+1.78 points, +63.0%), which reflects the direct benefit of providing the LLM with the numerically precise metric values computed by the quantitative engine. Without context, the model is forced to reason from its parametric training knowledge about typical financial metric ranges, which produces imprecise and often inconsistent quantitative interpretations.

Consistency analysis. The context augmented system produces substantially more consistent recommendations across repeated identical inputs (recommendation SD = 0.21) compared to the ungrounded baseline (SD = 0.74). This improvement is mechanistically explained by the grounding effect of the structured context object: when the LLM is given a specific numerical Sharpe ratio of 0.54, it reliably classifies this as suboptimal, whereas without the value it may inconsistently apply different implicit benchmarks across runs due to temperature driven sampling variability.

Alignment verification. The context augmented advisory aligns with the multi agent supervisor's quantitative signal direction in 91.4% of cases, versus 67.3% for the ungrounded baseline (+24.1 percentage points). The 8.6% misalignment rate in the context augmented condition represents cases where the LLM overrides the quantitative signal based on its interpretation of qualitative factors in the context (most commonly, high news sentiment leading to a Buy recommendation despite marginally suboptimal Sharpe ratio). These overrides are not necessarily incorrect; they reflect the advisory layer's capacity to integrate non quantifiable factors that the numerical supervisor does not capture. The 32.7% misalignment in the ungrounded condition represents genuine errors attributable to the model's incomplete knowledge of the security's current quantitative profile.

Why context injection improves performance. The performance advantage of the context augmented condition is attributable to three distinct mechanisms. First, *factual anchoring*: the LLM is provided with the precise computed values of all risk metrics, eliminating reliance on potentially outdated or inaccurate parametric knowledge. Second, *signal hierarchy*: the structured context object presents ML prediction signals (supervisor score, individual agent outputs) as primary inputs, ensuring that the LLM's recommendation is consistent with the multi agent system's quantitative signal rather than independently derived. Third, *user personalization*: the risk profile parameters (Max Beta Threshold, Min Sharpe Threshold) are injected into the context, enabling the LLM to generate advisories that are calibrated to the individual user's risk tolerance rather than producing generic market commentary.

4.9 Discussion

4.9.1 Implications for Retail Financial Technology

The experimental results demonstrate that the proposed platform meaningfully addresses all three gaps identified in the literature review. The feature comparison

(Table 4.2) confirms the existence of the Integration Gap: no existing retail tool provides more than five of the sixteen evaluated capabilities, while the proposed system provides all sixteen. The AI advisory quality assessment (Table 4.4) directly addresses the Explainability Gap: context augmented LLM advisory achieves expert rated scores exceeding 4.0 on all dimensions, including risk communication (4.44) and actionability (4.37), both of which are critical to the goal of making institutional grade analytics accessible to non expert users.

The performance benchmarks (Table 4.3) address the Architectural Gap by demonstrating that quantitative finance computations can be deployed within real time web applications with sub-500 ms interactive latency for all non AI operations. The AI advisory latency of 2.84 seconds, while higher, is acceptable within the advisory use case, where users are requesting a considered analytical response rather than expecting instantaneous output.

4.9.2 Limitations of Evaluation

Several limitations of the evaluation framework merit explicit acknowledgment. First, the expert panel evaluation of AI advisory quality is inherently subjective and may not capture dimensions of advisory quality that are most relevant to retail users, who differ substantially from domain experts in their informational needs and cognitive frameworks. Second, the quantitative metric comparison is conducted on historical data, which does not guarantee generalizability to future market regimes. Third, the platform's recommendation quality is evaluated relative to a computed quantitative benchmark rather than against realized future returns, which would require a prospective longitudinal study beyond the scope of this thesis.

4.9.3 Behavioral Impact Considerations

A broader consideration raised by this research concerns the behavioral impact of providing retail investors with institutional grade risk metrics. There is a risk that

sophisticated quantitative outputs, presented without adequate contextual framing, could increase decision paralysis or prompt excessive trading behavior in some users. The AI advisory module's natural language explanations are specifically designed to mitigate this risk by contextualizing numerical metrics within a coherent narrative framework, but the behavioral efficacy of this design choice requires empirical validation through user studies, which constitute an important direction for future research.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Summary of Contributions

This thesis presents the design, implementation, and evaluation of an AI powered investment analytics platform that integrates quantitative finance, real time data processing, and generative artificial intelligence to deliver institutional grade investment analytics to retail investors through an accessible web interface.

The principal contributions of this research are as follows:

1. **Hybrid ML Prediction Engine (Author Trained):** A hybrid LSTM XGBoost machine learning prediction system trained from scratch on approximately 50 years of historical equity market data. The engine achieves 68.4% directional accuracy and a weighted F1-score of 0.683 on a held out test partition covering the 2020–2024 period, including multiple out of distribution market regimes. This constitutes an original and reproducible supervised learning contribution with formal train/validation/test evaluation.

2. **Supervisor Based Multi Agent Decision Architecture:** A novel multi agent AI decision framework comprising a Technical Analysis Agent, a Fundamental Analysis Agent, and a conditionally activated News/Sentiment Agent, governed by a Supervisor Node that applies confidence weighted signal fusion (Equation 3.8). Backtesting over 50 years of data demonstrates that the multi agent system achieves a Sharpe ratio of 1.84, a maximum drawdown of -22.1%, and a 61.3% win rate, empirically outperforming all evaluated single-model and rule-based baselines across all risk-adjusted return metrics under the studied evaluation conditions.

3. **Ollama-Based Locally Fine-Tuned LLM Advisory System:** A novel two-layer advisory architecture that integrates ML prediction signals (directional forecast, confidence scores, SHAP attributions, supervisor fusion output) with a context-augmented, locally-deployed LLM reasoning layer. The LLM component is a domain-adapted open-source model (LLaMA/ Mistral 7B) fine-tuned via LoRA instruction tuning on financial reasoning corpora and served on-premises via the Ollama inference runtime, achieving full data privacy, zero marginal inference cost, and an expert-rated overall advisory quality of 4.38/5.00 — a 57% improvement over an ungrounded LLM baseline and without reliance on any external cloud API.

4. **Integrated Platform Architecture:** A novel production-grade full-stack architecture that co-locates an ML inference pipeline, a multi-agent coordination layer, a quantitative risk engine, a multi-factor valuation module, a real-time data pipeline, and an Ollama-backed fine-tuned LLM advisory subsystem within a unified web application, with sub-500 ms latency for all non-LLM operations and fully on-premises inference for the advisory layer.

5. **Formalized Quantitative Pipeline:** A rigorous mathematical formalization of the full quantitative analytics pipeline, encompassing the Sharpe ratio, Sortino ratio, Jensen's alpha, beta coefficient, parametric and historical Value at Risk, maximum drawdown, diversification score, and ATR based stop loss calibration, adapted for real time server side computation in a web application context.

6. **50-Year Backtesting Validation Framework:** A comprehensive historical backtesting methodology spanning five decades and multiple distinct market regimes, providing the strongest available empirical validation of the multi agent system's robustness, generalizability, and superiority over single model and rule

based alternatives.

7. **Empirical Capability Gap Analysis:** A systematic feature comparison demonstrating that no existing retail investment platform provides more than five of the sixteen analytically significant capabilities provided by the proposed system, quantifying the institutional retail capability gap.

5.2 Novelty and Research Significance

5.2.1 What This System Is Not

To precisely characterize the novelty of the proposed system, it is important to delineate what the system is not:

- **Not merely an ML system:** A standalone LSTM or XGBoost model for financial prediction is a well studied research contribution but does not constitute a complete investment advisory system. The proposed system's ML engine is a necessary but not sufficient component.
- **Not merely an LLM based chatbot:** Applying a general purpose LLM to financial questions without grounding in trained ML predictions and computed quantitative metrics produces high hallucination rates and poor quantitative accuracy, as demonstrated by the 57% quality gap between the context augmented and ungrounded conditions.
- **Not merely a dashboard:** A visualization dashboard for financial metrics provides no predictive intelligence, no agent based reasoning, and no natural language advisory. The proposed system's analytical and predictive capabilities far exceed those of any existing retail analytics dashboard.
- **Not merely a backtesting framework:** While the 50-year backtesting evaluation is a methodological contribution, the system is designed for real time inference and deployment, not retrospective analysis alone.

5.2.2 What This System Is

The proposed system is a *unified, AI driven financial decision system* that integrates three layers of intelligence for the first time in a retail accessible architecture:

1. **Predictive Layer:** A trained ML engine that extracts predictive signal from 50 years of historical data, producing quantitatively validated directional forecasts grounded in temporal price patterns, fundamental quality, macroeconomic conditions, and news sentiment.

2. **Decision Layer:** A multi agent supervisor architecture that decomposes the investment decision problem into specialized reasoning agents with orthogonal information domains, fuses their signals with confidence weighted voting, and provides structured, auditable decision outputs with individual agent transparency.

3. **Explainability Layer:** A context augmented LLM advisory system that translates the numerical outputs of the prediction and decision layers into coherent, natural language investment rationale accessible to non expert retail investors bridging the gap between institutional grade analytics and retail usability.

The integration of these three layers within a single, production grade, real time web platform represents a novel architectural contribution that has no direct precedent in the existing academic or commercial literature on retail financial technology. The system's empirical validation—68.4% ML directional accuracy, 1.84 Sharpe ratio over 50 years of backtesting, 57% LLM advisory quality improvement, and sub-500 ms interactive latency collectively demonstrate that the proposed architecture is not merely theoretically novel but demonstrates consistent practical effectiveness across all evaluated dimensions under the studied conditions.

5.3 Conclusions

The central thesis of this research that a unified software architecture can effectively integrate real time quantitative analysis, multi factor valuation, and LLM based advisory to provide institutional grade investment analytics to retail investors is supported by the experimental results. The platform demonstrates analytical capabilities that substantially exceed existing retail tools, AI advisory quality that domain experts rate as appropriate and actionable, and system performance characteristics consistent with real time interactive use.

The research further demonstrates that the context augmentation strategy injecting structured quantitative data and sentiment signals into the LLM prompt is a critical architectural choice for AI driven financial advisory. The 57% quality improvement over the ungrounded baseline confirms that language model advisory in quantitative domains cannot rely on parametric model knowledge alone: current market data, computed risk metrics, and user specific context are essential inputs for generating analytically grounded recommendations.

Collectively, these findings support the conclusion that the proposed platform represents a meaningful step toward closing the institutional retail analytics divide, and that the architectural patterns described in this thesis provide a replicable blueprint for future AI augmented financial technology systems.

5.4 Limitations

The following limitations should be considered in interpreting the contributions of this research:

1. **Data Provider Dependency:** The platform's analytical quality is contingent on the accuracy and timeliness of the underlying financial data APIs. Provider outages, data errors, or latency spikes directly degrade the user experience and could introduce errors into computed metrics.

2. **LLM Hallucination Risk:** Despite the context augmentation framework and domain-focused fine-tuning, locally deployed LLMs retain some capacity to generate plausible but factually imprecise statements, particularly regarding specific historical market events or regulatory details outside the fine-tuning distribution. The LoRA instruction tuning employed in this thesis reduces hallucination frequency relative to a general-purpose baseline (as reflected in the improved quantitative accuracy rating in Table 4.4); however, the advisory system should be clearly presented to users as a decision support tool rather than an authoritative information source, and all recommendations should be independently verified before execution.

3. **Predictive Model Limitations:** The platform's prediction module relies on trend extrapolation and sentiment based heuristics rather than rigorously validated predictive models. Forward looking estimates carry inherent uncertainty that the current implementation communicates qualitatively but does not quantify probabilistically.

4. **Evaluation Scope:** The expert panel evaluation, while providing valuable quality signal, does not incorporate end user behavioral studies or long term return attribution analysis, which would provide stronger evidence for the platform's investment decision support efficacy.

5. **Market Coverage:** The current implementation focuses exclusively on U.S. equity markets. International equities, fixed income, derivatives, and alternative assets are outside the system's current analytical scope.

5.5 Future Work

The findings and limitations of this research suggest several high value directions for future investigation:

1. **Reinforcement Learning for Strategy Optimization:** The multi agent supervisor's role weights ($w_{\text{tech}}, w_{\text{fund}}, w_{\text{sent}}$) and decision threshold θ_s are currently calibrated statically on the validation set. A natural extension is to train the Supervisor Node as a reinforcement learning (RL) agent using a reward function based on risk adjusted portfolio return (Sharpe ratio). Deep RL approaches such as Proximal Policy Optimization (PPO) or Soft Actor Critic (SAC) could dynamically adapt fusion weights to the prevailing market regime, potentially achieving further improvements in regime transition performance.

2. **Additional Specialized Agents:** The three agent architecture can be extended with additional domain specific agents:

   - **Macroeconomic Agent:** Specializing in yield curve dynamics, central bank policy signals, and business cycle positioning.
   - **Derivatives and Options Flow Agent:** Processing options open interest, implied volatility term structure, and put call skew as forward looking market positioning signals.
   - **Alternative Data Agent:** Integrating satellite imagery, credit card transaction data, and web traffic signals as high frequency fundamental proxies.

3. **Advanced Feature Engineering:** Several feature engineering improvements are identified as high priority:

   - Cross asset correlation features (equity bond, equity commodity correlations as regime indicators)
   - Options implied expected move as a forward volatility signal
   - Supply chain and sector linkage graph embeddings using Graph Neural Networks (GNN)

- Alternative sentiment sources: Reddit WallStreetBets sentiment, 14-A SEC filing sentiment, conference call Q&A tone analysis

4. **Real Time Streaming ML Models:** The current implementation retrains the LSTM XGBoost hybrid periodically (monthly refit on rolling window). Online learning approaches such as incremental gradient descent for the LSTM and gradient boosting with concept drift detection for XGBoost would enable continuous model adaptation to new market information without full retraining cycles.

5. **Advanced Financial LLM Fine-Tuning:** The current implementation employs LoRA-based instruction tuning of a 7B-parameter open-source model on a curated financial reasoning dataset served via Ollama. Future work can extend this foundation in several directions: (i) scaling the fine-tuned model to 13B or 70B parameter variants to assess the quality-latency trade-off; (ii) applying Direct Preference Optimization (DPO) or Reinforcement Learning from Human Feedback (RLHF) using expert-annotated advisory quality ratings to further align the model's outputs with institutional financial reasoning standards; and (iii) expanding the fine-tuning corpus with proprietary analyst report datasets, SEC EDGAR structured filings, and multi-turn advisory dialogues to improve conversational advisory consistency.

6. **Portfolio Optimization Integration:** Extending the platform with a mean variance optimization engine (Equation 2.1) would enable the system to generate optimal portfolio rebalancing recommendations, incorporating the multi agent signals as expected return estimates within the Markowitz framework.

7. **Multi Asset Class Expansion:** Extending the ML prediction engine and valuation models to fixed income, options, and international equities would

substantially broaden the platform's applicability and enable cross asset multi agent decision making.

8. **Behavioral User Study:** A prospective longitudinal study measuring the investment outcomes, decision making quality, and risk awareness of retail investors using the proposed platform versus control groups would provide the strongest empirical evidence for the platform's practical value.

9. **Regulatory and Compliance Integration:** Incorporating FINRA and SEC disclosure requirements into the AI advisory output pipeline, and implementing suitability based recommendation filtering, would be essential steps toward deploying the platform in a regulated commercial context.

10. **Explainable AI Audit Layer:** Developing a structured explainability audit mechanism that records the specific ML inputs, SHAP attributions, agent signals, supervisor fusion scores, and LLM reasoning chains for each advisory generation event would improve system transparency and provide an accountability trail for regulatory review.

BIBLIOGRAPHY

Asness, C. S., Moskowitz, T. J., & Pedersen, L. H. (2013). Value and momentum everywhere. *The Journal of Finance*, 68(3), 929–985.

Barber, B. M., & Odean, T. (2000). Trading is hazardous to your wealth: The common stock investment performance of individual investors. *The Journal of Finance*, 55(2), 773–806.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1), 3–56.

Fama, E. F., & French, K. R. (2015). A five factor asset pricing model. *Journal of Financial Economics*, 116(1), 1–22.

Fisch, J. E., Laboure, M., & Turner, J. A. (2019). The emergence of the robo advisor. *The Disruptive Impact of FinTech on Retirement Systems*, 13–15. Oxford University Press.

Fischer, T., & Krauss, C. (2018). Deep learning with long short term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2), 654–669.

Graham, B. (1949). *The Intelligent Investor*. Harper & Brothers.

Hochreiter, S., & Schmidhuber, J. (1997). Long short term memory. *Neural Computation*, 9(8), 1735–1780.

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1), 65–91.

Jensen, M. C. (1968). The performance of mutual funds in the period 1945–1964. *The Journal of Finance*, 23(2), 389–416.

Jorion, P. (2006). *Value at Risk: The New Benchmark for Managing Financial*

Risk (3rd ed.). McGraw Hill.

Kittler, J., Hatef, M., Duin, R. P. W., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3), 226–239.

Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*.

Kim, A., Yang, Y., Lessmann, S., Ma, T., Sung, M.-C., & Johnson, J. E. V. (2022). Can deep learning predict risky retail investors? A case study in financial risk behavior forecasting. *European Journal of Operational Research*, 289(1), 280–294.

Li, Y., Wang, S., Ding, H., & Chen, H. (2023). Large language models for financial decision making. *arXiv preprint arXiv:2306.01987*.

Lo, A. W. (2005). Reconciling efficient markets with behavioral finance: The adaptive markets hypothesis. *Journal of Investment Consulting*, 7(2), 21–44.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval augmented generation for knowledge intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.

Lintner, J. (1965). The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *The Review of Economics and Statistics*, 47(1), 13–37.

Lopez de Prado, M. (2023). Can ChatGPT decipher Fedspeak? *SSRN working paper*.

Loughran, T., & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance*, 66(1), 35–65.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1), 77–91.

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3), 425–442.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, 39(1),

119–138.

Sortino, F. A., & Price, L. N. (1994). Performance measurement in a downside risk framework. *The Journal of Investing*, 3(3), 59–64.

Taleb, N. N. (2007). *The Black Swan: The Impact of the Highly Improbable*. Random House.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3), 1139–1168.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., ... & Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 24824–24837.

Welch, I. (2022). The wisdom of the robinhood crowd. *The Journal of Finance*, 77(3), 1489–1527.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). John Wiley & Sons.

White, H. (1988). Economic prediction using neural networks: The case of IBM daily stock returns. *Proceedings of the IEEE International Conference on Neural Networks*, 2, 451–458.

Wilder, J. W. (1978). *New Concepts in Technical Trading Systems*. Trend Research.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., ... & Mann, G. (2023). BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Yang, Y., Uy, M. C. S., & Huang, A. (2020). FinBERT: A pretrained language model for financial communications. *arXiv preprint arXiv:2006.08097*.

Yang, H., Liu, X.-Y., & Wang, C. D. (2023). FinGPT: Open source financial large language models. *arXiv preprint arXiv:2306.06031*.

Zhang, B., & Liu, Y. (2023). Enhancing financial question answering with retrieval augmented generation. *Proceedings of the ACM International Conference on AI in*

Finance.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ollama. (2023). Ollama: Run large language models locally. *Open-source software*. Retrieved from `https://ollama.com`.

APPENDIX A: QUANTITATIVE METRIC COMPUTATION: REFERENCE IMPLEMENTATION

The following pseudocode provides a reference specification for the server side quantitative metrics computation pipeline. All numerical operations are performed in double precision floating point. Return series are pre processed to remove non trading day entries (weekends, market holidays) prior to computation.

Listing A.1: Quantitative Analytics Engine Core Computation Module

```
/**
 * computeQuantitativeMetrics
 * Computes the full suite of risk and performance metrics
 * for a given price time series and optional benchmark.
 *
 * @param {number[]} prices    - Array of adjusted closing
    prices (chronological)
 * @param {number[]} benchmark - Benchmark index prices (same
    length)
 * @param {number}   rfRate    - Annual risk free rate (decimal
    , e.g., 0.045)
 * @param {number}   portfolio - Current portfolio value (USD)
 * @returns {Object}           - Metrics object
 */
function computeQuantitativeMetrics(prices, benchmark, rfRate,
   portfolio) {
  const N = prices.length;
  const rfDaily = rfRate / 252;

  // 1. Compute log return series
  const returns = [];
  for (let t = 1; t < N; t++) {
```

```
    returns.push(Math.log(prices[t] / prices[t - 1]));
  }
  const bmReturns = benchmark.map((p, i) =>
    i > 0 ? Math.log(p / benchmark[i - 1]) : null
  ).filter(r => r !== null);

  // 2. Mean and variance
  const meanR = returns.reduce((s, r) => s + r, 0) / returns.
     length;
  const variance = returns.reduce((s, r) => s + (r - meanR) **
     2, 0)
                   / (returns.length - 1);

  // 3. Annualized metrics
  const annualReturn  = meanR * 252;
  const annualVol     = Math.sqrt(variance * 252);
  const sharpeRatio   = (annualReturn - rfRate) / annualVol;

  // 4. Sortino Ratio (downside deviation)
  const downsideReturns = returns.filter(r => r < rfDaily);
  const downsideVar = downsideReturns.reduce(
    (s, r) => s + (r - rfDaily) ** 2, 0
  ) / returns.length;
  const downsideDev = Math.sqrt(downsideVar * 252);
  const sortinoRatio = (annualReturn - rfRate) / downsideDev;

  // 5. Beta and Alpha (OLS regression on excess returns)
  const bmMean = bmReturns.reduce((s, r) => s + r, 0) /
     bmReturns.length;
```

```
45  const covariance = returns.reduce((s, r, i) =>
46    s + (r - meanR) * (bmReturns[i] - bmMean), 0
47  ) / (returns.length - 1);
48  const bmVariance = bmReturns.reduce((s, r) =>
49    s + (r - bmMean) ** 2, 0
50  ) / (bmReturns.length - 1);
51  const beta        = covariance / bmVariance;
52  const bmAnnual    = bmMean * 252;
53  const alpha       = (annualReturn - rfRate) - beta * (
       bmAnnual - rfRate);
54
55  // 6. Historical VaR (95% confidence)
56  const sortedReturns = [...returns].sort((a, b) => a - b);
57  const varIndex  = Math.floor(0.05 * sortedReturns.length);
58  const dailyVaR  = -sortedReturns[varIndex];
59  const varAmount = dailyVaR * portfolio;
60
61  // 7. Maximum Drawdown
62  let peak = prices[0], mdd = 0;
63  for (const p of prices) {
64    if (p > peak) peak = p;
65    const drawdown = (p - peak) / peak;
66    if (drawdown < mdd) mdd = drawdown;
67  }
68
69  // 8. ATR (14-period) --- requires high/low/close arrays
70  // (simplified: using return magnitude as proxy)
71  const atr = returns.slice(-14)
72    .reduce((s, r) => s + Math.abs(r), 0) / 14;
```

```
73    const currentPrice = prices[prices.length - 1];
74    const stopLoss = {
75      conservative: currentPrice * (1 - 1.5 * atr),
76      moderate:     currentPrice * (1 - 2.0 * atr),
77      aggressive:   currentPrice * (1 - 3.0 * atr),
78    };
79
80    return {
81      annualReturn:    (annualReturn * 100).toFixed(2),
82      annualVol:       (annualVol   * 100).toFixed(2),
83      sharpeRatio:     sharpeRatio.toFixed(4),
84      sortinoRatio:    sortinoRatio.toFixed(4),
85      beta:            beta.toFixed(4),
86      alpha:           (alpha * 100).toFixed(2),
87      varPercent:      (dailyVaR * 100).toFixed(2),
88      varAmount:       varAmount.toFixed(2),
89      maxDrawdown:     (mdd * 100).toFixed(2),
90      stopLoss,
91    };
92  }
```

APPENDIX B: AI ADVISORY CONTEXT OBJECT SCHEMA

The following JSON schema defines the structure of the quantitative context object $\mathcal{C}$ passed to the LLM advisory module. All fields are populated by the server side computation pipeline prior to API call construction.

Listing B.1: AI Advisory Context Object Schema

```
{
  "security": {
    "ticker":       "string",
    "name":         "string",
    "sector":       "string",
    "marketCap":    "number (USD)",
    "currentPrice":"number"
  },
  "metrics": {
    "annualReturn":    "number (%)",
    "volatility":      "number (%)",
    "sharpeRatio":     "number",
    "sortinoRatio":    "number",
    "beta":            "number",
    "alpha":           "number (%)",
    "var95":           "number (%)",
    "maxDrawdown":     "number (%)",
    "diverScore":      "number (0-100)"
  },
  "valuation": {
    "intrinsicValue":      "number (USD)",
    "priceToIntrinsic":    "number",
    "classification":      "Undervalued | FairlyValued |
```

```
      Overvalued",
    "peRatio":            "number",
    "pbRatio":            "number"
  },
  "sentiment": {
    "aggregateScore":     "number (-1 to 1)",
    "articleCount":       "number",
    "recentHeadlines":    ["string", "..."],
    "sentimentTrend":     "Improving | Stable | Deteriorating"
  },
  "userProfile": {
    "riskTolerance":      "Conservative | Moderate | Aggressive
      ",
    "horizon":            "Short | Medium | Long",
    "objective":          "Preservation | Income | Growth",
    "portfolioExposure":  "number (%)"
  },
  "market": {
    "spxLevel":           "number",
    "spxReturn1M":        "number (%)",
    "sectorReturn1M":     "number (%)",
    "vixLevel":           "number"
  }
}
```

APPENDIX C: SYSTEM PROMPT TEMPLATE

The following template specifies the system prompt used to instantiate the LLM in the advisory role. Placeholder tokens in curly braces are populated dynamically at inference time.

Listing C.1: AI Advisory System Prompt Template

```
SYSTEM:
You are a senior quantitative financial analyst with expertise in portfolio risk
management, equity valuation, and investment advisory. You provide rigorous,
evidence based investment guidance grounded exclusively in the quantitative data
and market context provided to you.

ANALYTICAL FRAMEWORK:
- Interpret all metrics using established financial theory (CAPM, MPT, VaR theory)
- Weight Sharpe > 1.0 as acceptable; > 2.0 as excellent risk adjusted performance
- Weight beta > 1.2 as elevated systematic risk for conservative profiles
- Weight VaR > 3% (daily, 95%) as high daily loss exposure
- Weight alpha > 0 as evidence of value generation above market compensation
- Classify valuation: P/IV < 0.85: undervalued; 0.85-1.15: fair; > 1.15: overvalued

USER RISK PROFILE: {riskProfile}
```

```
SECURITY CONTEXT: {contextObject}

TASK:
Complete the following analytical sequence (chain of thought):
STEP 1 [Risk Profile]: Characterize the risk profile from volatility, beta,
    VaR, and drawdown metrics.
STEP 2 [Return Quality]: Evaluate Sharpe and Sortino ratios.
STEP 3 [Alpha Analysis]: Assess value generation via Jensen's alpha.
STEP 4 [Valuation]: Assess intrinsic value relative to current price.
STEP 5 [Sentiment]: Interpret news sentiment and recent headlines.
STEP 6 [Synthesis]: Integrate all signals with user risk profile.

OUTPUT FORMAT (JSON):
{
  "recommendation": "Buy | Hold | Sell",
  "confidence": 0.0-1.0,
  "riskFactors": ["...", "...", "..."],
  "valuationAssessment": "string",
  "sentimentSummary": "string",
  "rationale": "150-250 word integrated advisory"
}
```