Explainable Stock Recommendation System

Background

In contemporary financial markets, investors rely heavily on digital platforms such as Bloomberg Terminal, Yahoo Finance, and StockTwits to obtain data-driven insights. While these platforms each fulfill specialized roles—Bloomberg Terminal in delivering detailed financial fundamentals, StockTwits in reflecting market sentiment through social media trends, and robo-advisory tools in generating strategic suggestions—they often function in silos and fail to offer a holistic analytical framework. Moreover, many existing systems lack sufficient interpretability, leaving users with recommendations that are difficult to evaluate and justify.

The fragmented nature of these services presents significant challenges for investors, who must navigate disparate sources to piece together comprehensive insights. Also, recent studies (Arrieta et al., 2020; Černevičienė & Kabašinskas, 2024) emphasize the necessity for transparency and explainability in financial decision-support systems, which is an imperative that is particularly critical in high-stakes investment contexts and a central focus of this course on Explainable AI.

Motivation for System Design

This recommendation system is designed to address the limitations of conventional financial platforms by unifying real-time sentiment analysis, fundamental financial evaluation, and technical strategy testing into a single, coherent framework. A distinguishing feature of the system is its robust integration of explainability analysis. By employing methods such as gradient-based attribution, attention mechanism visualization, and feature contribution quantification (e.g., Integrated Gradients, Self-Attention, and TreeSHAP), the system not only provides structured, evidence-based recommendations but also elucidates the rationale behind them. This transparency is intended to enhance user trust and facilitate informed decision-making.

Moreover, the system is developed with an emphasis on accessibility. Unlike many mainstream platforms that impose registration requirements and subscription fees to access detailed analyses, this platform is freely available and does not mandate user registration. All primary analytical modules, including sentiment, financial, and strategy evaluations, are offered at no cost. The final recommendation synthesis, serves as an enhancement rather than a prerequisite, ensuring that comprehensive insights remain accessible even without it.

By reducing access barriers and emphasizing transparency, this platform seeks to provide a comprehensive and interpretable investment support tool tailored to both novice and experienced users.

Stock Recommendation System

1. System Overview

The system we developed is a stock recommendation platform that provides investment suggestions based on multiple analytical approaches. The system displays key stock information such as the current price, percentage change, opening/high/low prices, and trading volume. It also features an interactive chart showing daily price data over the past year. A news feed is available to present the latest stock-related headlines, summaries, sources, and publication times. There

are three major analysis modules integrated into the system. The sentiment analysis module leverages news and social media data to assess market sentiment. The research analysis module focuses on financial reports and indicators. The strategy analysis module evaluates trading strategies with backtesting charts. Each analysis module comes with an explanatory button that provides detailed insights to help users understand the results. Multiple interpretability techniques are incorporated, including Integrated Gradient, Self Attention, TreeSHAP, and Cosine Similarity. The system also supports interpretability features for models such as Mistral and FinBERT. The final output is an investment recommendation rating (STRONG BUY, BUY, HOLD, SELL, STRONG SELL), accompanied by a thorough explanation and supporting rationale.

2. Core Business Logic

The data flow begins when a user inputs a stock ticker. The system retrieves its basic information and relevant news. Users can choose to run any of the three analysis modules. Results are stored in localStorage and used to generate the final investment recommendation. For the analysis process, the sentiment analysis module evaluates news content and generates sentiment scores and classifications. The financial analysis module assesses financial indicators and produces a financial health report. The strategy analysis module runs backtests, reviews strategy performance, and provides actionable recommendations. The recommendation is generated by consolidating the outcomes from all three analyses. An API determines the final rating based on the weighted importance of different factors, with an accompanying explanation for the recommendation. Interpretability features offer transparency into the model's decision-making process. They highlight the contribution of each analytical factor and use various visualizations to help users understand the final recommendation.

3. System Architecture and Technology Stack

The frontend is built using Next.js 14, React 18, and TypeScript. Tailwind CSS handles styling, and Chart.js is used for rendering charts. The application follows a component-based React structure. The backend utilizes a Flask API server powered by Python-based machine learning and NLP models. It also integrates various AI interpretability techniques and supports asynchronous task execution. The system connects to external APIs such as AlphaVantage for stock price and fundamental data, news APIs for fetching related news articles, and Hugging Face models for sentiment analysis and text processing.

4. Error Handling and Service Quality Assurance

To ensure API reliability, a key rotation mechanism is implemented to prevent limitations from single API keys. Rate limiting and queue management are used to comply with API provider policies. A multi-tier degradation strategy is in place. Level 1 switches to a backup model if the primary model fails. Level 2 employs cached data and rule-based engines when all models are unavailable. Any results from fallback mechanisms are clearly labeled to maintain transparency. Model analysis results are cached for four hours, but users can force-refresh the data at any time to retrieve the latest results.

5: Functional and Logical Breakdown for Key modules1: Sentiment Analysis

This section provides an in-depth explanation of one of the core components of the stock recommendation system: sentiment analysis. The sentiment analysis module is designed to evaluate the emotional tone conveyed in news articles and media content related to specific stocks.

5.1 Overall Workflow

The module begins by retrieving the latest news articles related to the target stock using financial news APIs. These news texts, including titles and summaries, undergo preprocessing to remove noise and ensure relevance. Afterward, the system calls FinBERT—a BERT variant specialized in financial sentiment analysis, hosted on Hugging Face—via a backend API. Each article is classified as Positive, Negative, or Neutral, with an accompanying confidence score. These results are used to calculate an overall sentiment score and generate a corresponding investment recommendation.

5.2 Data Sources and Text Processing

For sourcing data, the system leverages APIs like Finnhub and AlphaVantage to pull news from the past 30 days, ensuring the analysis reflects recent market sentiment. Preprocessing steps include text extraction, formatting titles and summaries into a unified string, removing empty or duplicate records, and retaining the five most recent valid entries. Special characters and formatting noise are also cleaned from the text.

5.3 AI Models and Analytical Workflow

The core model for sentiment analysis is FinBERT (model ID: ProsusAI/finbert), with a backup alternative (model ID: yiyanghkust/finbert-tone), both accessed through Hugging Face inference endpoints via backend API. The classification process assigns a sentiment label and confidence to each news item. Scores are attributed based on polarity: positive sentiment is assigned a positive value, negative sentiment a negative value, and neutral sentiment a score near zero. The system then computes an average score to determine the overall sentiment. Based on this average sentiment score, the module determines a collective emotional trend: greater than 0.3 signifies positive, less than -0.3 indicates negative, and scores in between are classified as neutral. It identifies the strongest positive and negative contributions and formulates an investment recommendation: Buy for positive, Sell for negative, and Hold for neutral sentiment.

5.4 Explainability System

Explainability features allow users to interpret the reasoning behind the sentiment results, highlighting influential factors in the decision-making process. These mechanisms are embedded in the module to ensure transparency and trust in the system. Here we use two explainable methods, one is Integrated Gradient and the other is Self Attention

5.4.1 Interpreting FinBERT Sentiment Predictions Using Integrated Gradients

In our approach, we employ the Integrated Gradients (IG) method to elucidate how individual tokens contribute to the sentiment score generated by the FinBERT model. This method provides a principled way to attribute the final prediction to the input features by quantifying their cumulative impact along a path from a baseline input to the actual input.

5.4.1.1 Method Overview

The key idea behind Integrated Gradients is to measure the sensitivity of the model's output with respect to each input feature as we gradually transition from a baseline to the actual input. The contribution of each token is obtained by multiplying the total change in its embedding by the average gradient of the model's output along this path.

5.4.1.2 Mathematical Formulation

Let f denote the output function of the FinBERT model, and let x be the input embedding vector for a specific token. We choose a baseline embedding x' (a zero vector) that represents the absence of information. The Integrated Gradient for the i-th component of the input is defined as:

$$\mathrm{IG}_i(x) = (x_i - x_i') \times \int_0^1 \frac{\partial f\big(x' + \alpha(x - x')\big)}{\partial x_i}\, d\alpha$$

In practice, this integral is approximated using a Riemann sum over N discrete steps:

$$\mathrm{IG}_i(x) \approx (x_i - x_i') \times \frac{1}{N} \sum_{k=1}^{N} \frac{\partial f\big(x' + \frac{k}{N}(x - x')\big)}{\partial x_i}$$

This formulation captures the cumulative effect of the token's contribution as its embedding transitions from the baseline x' to the actual embedding x.

5.4.1.3 Implementation in FinBERT

For sentiment analysis with FinBERT, the process is applied to each token in an input sentence. The key steps are as follows:

1. **Tokenization and Embedding Extraction:**
   The input sentence is tokenized, and each token is mapped to its corresponding embedding Einput through the model's embedding layer.
2. **Baseline Definition:**
   A baseline embedding Ebaseline is defined (a zero vector). The difference, ΔE = Einput − Ebaseline, quantifies the change from a non-informative state to the actual token representation.
3. **Interpolation Path Construction:**
   We construct a linear path between Ebaseline and Einput using:

   $$E(\alpha) = E_{\text{baseline}} + \alpha \times \Delta E, \quad \text{for } \alpha \in [0, 1]$$

   The path is discretized into N steps (e.g., 25 steps), where each step corresponds to a value

   $$\alpha_k = \frac{k}{N}$$

4. **Gradient Computation:**

   At each interpolation step alpha_k, the model computes the output score and the corresponding gradient:

   $$\frac{\partial f\big(E(\alpha_k)\big)}{\partial E}$$

This gradient indicates the sensitivity of the output to changes in the token's embedding at that particular point.

5. **Aggregation of Attributions:**
   The Integrated Gradient for the token is then approximated as:

$$\text{Attribution} \approx \Delta E \times \frac{1}{N} \sum_{k=1}^{N} \frac{\partial f\big(E(\alpha_k)\big)}{\partial E}$$

This product effectively captures the total contribution of the token by combining its intrinsic change (ΔE) with the model's average sensitivity.

5.4.1.4 Interpretation

The computed attributions for each token indicate how much each one influences the final sentiment prediction. A higher attribution suggests that slight variations in the token's embedding would lead to significant changes in the output, thereby highlighting its importance in the sentiment analysis.

By applying this method to all tokens in a sentence, we can aggregate the individual contributions to obtain an interpretable explanation of how the FinBERT model arrives at its overall sentiment score.

5.4.2 Self-Attention for Token Relationship Analysis

5.4.2.1 Overview

In this section, we explain how the self-attention mechanism is utilized within the FinBERT model to elucidate the relationships between tokens. By extracting the attention matrix from the final Transformer layer, we obtain a quantitative representation of inter-token interactions. This facilitates an in-depth understanding of how the model allocates focus across different parts of the input sequence.

5.4.2.2 Mathematical Formulation

Given an input sequence of tokens, each token xi is embedded into a continuous vector space, resulting in:

$$\mathbf{X} = [x_1, x_2, \ldots, x_n]$$

These embeddings are projected into three distinct representations—queries Q, keys K, and values V—using learnable weight matrices:

$$Q = XW^Q, \quad K = \mathbf{X}W^K, \quad V = \mathbf{X}W^V$$

The attention scores are computed using the scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where dk is the dimensionality of the key vectors. The softmax function ensures that the scores are normalized, thereby forming a probability distribution that quantifies the contribution of each token to the context of others.

In a multi-head configuration, suppose there are H heads; each head computes its own attention matrix A^{(h)}. These matrices are then averaged to yield a consolidated attention matrix:

$$A = \frac{1}{H} \sum_{h=1}^{H} A^{(h)}$$

where

$$A \in \mathbb{R}^{n \times n}$$

and each element A{ij} represents the attention weight from token i to token j.

5.4.2.3 Implementation Details

Within our implementation for FinBERT, the following procedure is applied:

1. **Final-Layer Selection:**
   The model is configured to output attention weights by enabling the flag for attention extraction. From the returned outputs, the attention matrix from the final layer is selected as it typically captures the richest semantic relationships.
2. **Averaging Across Heads:**
   To mitigate variability across individual attention heads, the attention matrices from all heads in the final layer are averaged.
3. **Interpretation of the Matrix:**
   Each entry A{ij} in the averaged matrix quantifies the degree to which token i attends to token j. This matrix is then used to analyze the importance and correlation of tokens within the text.

5.4.2.4 Interpretation and Insights

The resultant attention matrix provides insights into the internal workings of FinBERT. By examining the matrix, one can infer:

1. **Dominant Tokens:** Tokens that exhibit high attention weights across the sequence, indicating their significant role in model inference.
2. **Contextual Dependencies:** Patterns in the matrix reveal how contextual information is propagated, especially in cases where long-distance dependencies are critical.

5.5 Frontend Implementation Details

The frontend highlights sentiment using color coding—green for positive, gray for neutral, and red for negative—alongside the sentiment score and confidence level. It also displays a list of analyzed news entries with corresponding sentiment tags. To enhance user understanding, the interface provides status indicators showing API success rates, flags simulated data when fallbacks are used, and includes a debug panel.

5.6 Integration with the Broader System

The sentiment analysis results are stored in local storage and made available to other system components. They are broadcast through a custom event system to notify components of updates and serve as a critical input to the final investment recommendation.

6: Functional and Logical Breakdown for Key modules2: Financial Analysis

This section provides an in-depth explanation of one of the core components of the stock recommendation system: the Financial Analysis module. This module is designed to evaluate the financial health and performance of companies through structured financial key metrics data and AI-powered insights.

6.1 Overall Workflow

The financial analysis module begins by fetching basic and detailed financial data through various financial data APIs. The retrieved data is then preprocessed and structured into a standardized format. A request is made to the Mistral-7B-Instruct model hosted on Hugging Face to conduct a professional financial analysis. The system parses the model's output and extracts structured insights. TreeSHAP and light weight NLP method are then applied to provide explainability for the model's decision-making process. Finally, the financial analysis results are integrated into the overall recommendation logic of the system.

6.2 Data Sources and API Integration

To gather financial data, the system utilizes multiple APIs: Alpha Vantage provides company fundamentals, income statements, and balance sheet data. Financial Modeling Prep (FMP) supplies key financial ratios and performance indicators. Finnhub contributes analyst ratings and target prices. An unofficial Yahoo Finance API serves as an auxiliary data source for real-time stock information. Key metrics collected and analyzed include revenue growth, net income growth, current ratio, debt to equity, and return on equity.

6.3 AI Models and Analytical Workflow

The core model used for financial analysis is Mistral-7B-Instruct, a 7-billion-parameter instruction-tuned large language model hosted on Hugging Face (model ID: mistralai/Mistral-7B-Instruct-v0.1). The system constructs well-structured prompts that contain company details and financial indicators, guiding the model to generate content in five sections: financial summary, strengths, weaknesses, indicator analysis, and investment recommendation. These prompts explicitly instruct the model to deliver actionable recommendations (Buy, Hold, or Sell) and justify its conclusions.

6.4 Explainability System

To enhance interpretability, the module incorporates both textual and algorithmic techniques. Lightweight NLP methods highlight and interpret key sentences in the model's output, with the highest-ranked sentences being selected for display. TreeSHAP is applied to measure each financial indicator's contribution to the recommendation by contribution bar plots.

6.4.1 TreeSHAP for Mistral Analysis

6.4.1.1 Overview

To interpret the complex outputs of the Mistral model, we adopt a multi-step approach that combines perturbation-based sampling, a proxy modeling strategy using XGBoost, and the application of TreeSHAP for feature attribution.

6.4.1.1 Perturbation Sample Generation

In order to interpret the predictions of the Mistral model, we first generate a set of perturbation samples. The goal is to cover the local input space around the original financial data. The process is divided into two stages:

a. Baseline Preparation and Extreme Value Detection

- **Baseline Data:**
  We begin by copying the original financial data (denoted as *base_data*).

- **Extreme Value Identification:**
  For key indicators such as Return on Equity (ROE), Price-to-Earnings (P/E) ratio, and Net Income Growth, extreme values are detected if they exceed predefined thresholds. These extreme values trigger a modified perturbation strategy to ensure that the sampling captures critical edge cases.

b. Definition of Perturbation Strategies

For each financial feature (e.g., revenue growth, net income growth, current ratio, debt-to-equity, ROE, P/E ratio), we define the following strategies:

- **Absolute Perturbation:**
  Replace the original value with a value randomly selected from a predefined list (e.g., revenue growth might take values such as -30, -20, ..., 50).
- **Relative Perturbation:**
  Scale the original value by a randomly chosen multiplicative factor (e.g., 0.5, 1.0, 1.5), provided the original value is non-zero. If the original value is zero, the absolute strategy is used.
- **Extreme Perturbation:**
  With a certain probability, assign a value from a specified extreme range. For features already identified as extreme, the perturbation range is adjusted to include even more divergent values, potentially also inverting the sign or using a fraction of the original extreme value.

Additionally, each feature is assigned a probability of using the absolute perturbation; constraints such as non-negativity or minimum value limits; an importance weight to influence the likelihood of being selected for multi-feature perturbations.

c. Two-Phase Sampling

- **Single-Feature Perturbation:**
  For each feature, at least a minimum number of samples (e.g., 12 per feature) are generated by perturbing only that specific feature while keeping all others at their baseline values. This isolates the impact of individual features.
- **Multi-Feature Perturbation:**
  The remaining samples are generated by perturbing multiple features simultaneously. In the initial set of samples, more extreme combinations are prioritized by selecting 2–3 features with a higher probability for extreme perturbation. For subsequent samples, features are chosen based on their importance weights, ensuring that key features (e.g., ROE, net income growth) are frequently included.

Each perturbation sample is then fed to the Mistral API to obtain a corresponding prediction score, and the sample's details (including original and perturbed values, perturbation type, and resulting score) are recorded for subsequent model training.

6.4.1.2 Rationale for Using an XGBoost Proxy Model

Directly explaining the Mistral model is challenging due to its nature as a complex, black-box language model. To address this, we adopt an XGBoost proxy model for several reasons:

- **Interpretability:**
  XGBoost, a tree-based model, offers structural transparency and can be efficiently explained using TreeSHAP. Its tree ensemble structure allows for exact computation of feature contributions.

- **Efficiency:**
  Unlike the Mistral model, which is accessed via API calls and may incur latency or reliability issues, XGBoost can be trained and evaluated rapidly. Once trained on the perturbation samples with predictions from the Mistral model, it approximates the behavior of the Mistral model within the input space.
- **Consistency with SHAP Framework:**
  TreeSHAP is specifically designed for tree models. By training an XGBoost model on the generated samples, we ensure that the SHAP explanations are both accurate and computationally efficient.

6.4.1.3 TreeSHAP: Theory and Application

a. Shapley Values in Cooperative Game Theory

The Shapley value is a concept from cooperative game theory used to determine the contribution of each player (feature) in a game (model prediction). For a model f and a given input x with feature set N, the Shapley value for feature i is defined as:

$$\phi_i(f, x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f_x(S \cup \{i\}) - f_x(S)]$$

Here, S represents a subset of features excluding i, and fx(S) denotes the expected model output conditioned on the feature subset S.

b. TreeSHAP's Efficient Computation

For tree models, the straightforward computation of Shapley values is computationally prohibitive due to the need to evaluate every possible subset S. TreeSHAP leverages the structure of decision trees to achieve polynomial time complexity. Its main components include:

- **Path-Based Decomposition:**
  Each decision tree splits based on a single feature. TreeSHAP traverses each path from the root to a leaf, tracking the fraction of training data (the "coverage") that passes through each node.
- **Dynamic Programming:**
  The algorithm accumulates contributions along the decision paths without enumerating all possible subsets. For a single tree, the model output is represented as:

  $$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i$$

  where ϕ0 is the baseline and ϕi are the individual feature contributions. For an ensemble of trees (e.g., XGBoost):

  $$f(x) = \sum_{t=1}^{T} \left(\phi_{0,t} + \sum_{i=1}^{M} \phi_{i,t} \right)$$

  The final Shapley value for feature i is obtained by summing its contributions across all trees:

  $$\phi_i = \sum_{t=1}^{T} \phi_{i,t}$$

This method reduces the computational complexity significantly to O(TLD^2) (with T trees, L leaves per tree, and maximum depth D).

c. Application to the XGBoost Model

In our context, after training the XGBoost proxy model using the perturbation samples, we apply TreeSHAP to explain the model's predictions:

- **Explainer Construction:**
  We instantiate a TreeSHAP explainer using the XGBoost model:

  explainer = shap.TreeExplainer(model)

- **Global and Local Explanations:**
  - **Global SHAP Values:**
    We compute SHAP values for all perturbation samples, take the average of the absolute SHAP values per feature, and normalize these values to obtain a global feature importance ranking.
  - **Local SHAP Values (Used to present in final visualization):**
    For the original financial data point, we calculate individual SHAP values which indicate how each feature shifts the prediction relative to the model's baseline.

In summary, the explanation logic for the Mistral model is achieved through a three-step process. This approach combines robust sampling, a transparent proxy model, and advanced interpretability techniques to deliver a comprehensive explanation of the original model's output.

6.4.2 Semantic Similarity Analysis for Highlighting Key Sentences in Investment Recommendations

6.4.2.1 Overview

Our methodology integrates global semantic similarity analysis with domain-specific keyword weighting to identify critical sentences in investment recommendation reports. The strategy is comprised of the following steps.

6.4.2.2 Preprocessing and Sentence Segmentation

To avoid erroneous splitting of numerical values (e.g., "12.5%"), a regular expression is applied to detect numeric patterns of the form digit. These decimal points are temporarily replaced with a unique placeholder. After protecting the decimals, the text is segmented into sentences using punctuation marks (e.g., periods, exclamation points, and question marks). Once segmentation is complete, the placeholders are reverted back to actual decimal points.

6.4.2.3 Sentence Embedding and Semantic Centrality Computation

- **Embedding Generation:**
  Each sentence is transformed into a high-dimensional vector using a pre-trained sentence transformer model (paraphrase-MiniLM-L6-v2). This vector representation captures the semantic content of the sentence.
- **Cosine Similarity:**
  The pairwise cosine similarity between sentence vectors vi and vj is computed using the formula:

  $$\text{cosine\_sim}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \, \|\mathbf{v}_j\|}$$

  To ensure non-negative values, any negative similarity scores are clamped to zero.
- **Centrality Score:**
  The semantic centrality Ci of each sentence i is defined as the average cosine similarity of that sentence with all other sentences:

$$C_i = \frac{1}{N} \sum_{j=1}^{N} \max(0, \text{cosine\_sim}(\mathbf{v}_i, \mathbf{v}_j))$$

where N is the total number of sentences. A higher centrality score indicates that the sentence is more representative of the overall semantic content of the document.

6.4.2.4 Domain-Specific Keyword Weighting

- **Lexicon-Based Weighting:**
  A set of financial indicators, prompt words, and special phrases is used to assess the importance of each sentence from a domain perspective. Each occurrence of a financial indicator or prompt word contributes a specific incremental weight. Special phrases are assigned a higher multiplier to reflect their critical importance.
- **Numerical Influence:**
  The presence of numeric values (including percentages) is also taken into account, with each numerical occurrence contributing a fixed incremental weight (e.g., 0.2 per number).
- **Normalization:**
  The aggregate keyword weight for each sentence is normalized relative to the maximum observed weight across all sentences, ensuring comparability.

6.4.2.5 Combined Scoring and Ranking

- **Score Integration:**
  The final importance score Si for each sentence is derived by combining the semantic centrality and the normalized keyword weight. Specifically, the score is computed as:

  $$S_i = \alpha\, C_i + (1 - \alpha)\, W_i$$

  where Ci is the normalized semantic centrality score; Wi is the normalized keyword weight; alpha is set to 0.6, placing greater emphasis on the semantic centrality.
- **Sentence Ranking:**
  After computing the final scores, the sentences are sorted in descending order. The highest-ranked sentences are selected as the key sentences for display.

This integrated approach leverages both global semantic context and specialized financial terminology to deliver a reliable extraction of pivotal sentences, enhancing the clarity and impact of investment recommendations.

6.5 Frontend Implementation Details

Results are displayed with intuitive visual cues: green for Buy, blue for Hold, and red for Sell. Users can expand or collapse content for readability. Interactive TreeSHAP visualizations allow users to explore the influence of various financial metrics on the model's recommendation. Semantic similarity analysis is used to identify and highlight key sentences in investment recommendations, with the highest-ranked sentences being selected for display.

6.6 Integration with the Broader System

Financial analysis results serve as a critical input alongside sentiment and strategy analyses. All three modules communicate updates through a custom event-driven system. Their outputs are aggregated into the final recommendation engine, where a comprehensive analysis is conducted to produce actionable investment advice.

7: Functional and Logical Breakdown for Key modules3: Strategy Analysis

This section provides an in-depth explanation of one of the core components of the stock recommendation system: the Strategy Analysis module. This module is designed to evaluate historical stock data using technical indicators and trading strategies to generate actionable investment signals.

7.1 Overall Workflow

The strategy analysis module follows a structured process. It begins by acquiring historical price and volume data for the target stock. Various technical analysis strategies and indicators are then applied. The system performs backtesting to evaluate historical performance and potential returns. Based on the results, it generates trading signals and recommendations derived from technical analysis. Clear explanations are provided to help users understand the rationale behind each strategy suggestion. Finally, the outputs are integrated into the system's overall investment recommendation engine.

7.2 Data Sources and Technical Indicators

To obtain historical OHLC (Open, High, Low, Close) data, the module uses APIs from Alpha Vantage and Yahoo Finance. The default time frame is one year of daily data. The key technical indicators calculated include simple and exponential moving averages (SMA and EMA) with various periods (e.g., 5, 10, 20, 50, 200 days), the relative strength index (RSI) with a default 14-day setting, the moving average convergence divergence (MACD) using standard parameters (12, 26, 9), Bollinger Bands with a 20-day moving average and two standard deviations, volume-weighted average price (VWAP), and average true range (ATR) as a volatility metric.

7.3 Trading Strategies and Backtesting

The module simulates trading strategies on historical data to assess their effectiveness. It evaluates key performance metrics such as annualized return, maximum drawdown, Sharpe ratio, and win rate. All historical signals are logged, including the trigger date, price, and condition. A capital curve simulates portfolio growth over time under the selected strategy, providing a visual representation of past performance. Below is a detailed description of each strategy.

7.3.1 Moving Average Strategy

Core Principle:

- Calculate 20-day and 50-day moving averages.
- Generate a **buy signal** when the 20-day moving average crosses upward through the 50-day moving average (a "golden cross").
- Generate a **sell signal** when the 20-day moving average crosses downward through the 50-day moving average (a "death cross").
- This approach is geared toward capturing medium- to long-term trend changes.

7.3.2 RSI (Relative Strength Index) Strategy

Core Principle:

- Compute a 14-day RSI.

- Generate a **buy signal** when RSI rises from below 30 (indicating a move out of oversold conditions).
- Generate a **sell signal** when RSI falls from above 70 (indicating a move out of overbought conditions).
- This strategy focuses on identifying potential reversal points in oscillating markets.

7.3.3 Bollinger Bands Strategy

Core Principle:

- Calculate a 20-day moving average and construct bands using two standard deviations above and below the moving average.
- Generate a **buy signal** when the price touches the lower band and subsequently rebounds.
- Generate a **sell signal** when the price reaches the upper band and then declines.
- This strategy is designed to capture reversal opportunities within price fluctuation ranges.

7.3.4 Trend Following Strategy

Core Principle:

- Combine multiple technical indicators, including moving averages, RSI, and price trends.
- **Buy Signal Conditions:**
  - The short-term moving average is positioned above the long-term moving average.
  - RSI is above 50, suggesting market strength.
  - The current price is higher than the price from five days prior.
- **Sell Signal Conditions:**
  - The inverse of the above conditions.
- Additionally, the strategy incorporates logic to avoid generating overly frequent signals, thereby reducing noise and potential false signals.
- This approach is focused on capturing medium- to long-term trends while minimizing short-term fluctuations.

7.3.4 Backtrader Macro Strategy

Core Principle:

- Integrate technical analysis with macroeconomic data.
- Evaluate the current economic environment using indicators such as inflation, GDP growth, interest rates, and unemployment. Based on these, classify the market environment into states like EXPANSION, OVERHEATING, SLOWDOWN, RECESSION, or RECOVERY.
- Adjust technical trading signals according to the assessed macroeconomic environment to better capture market cycles and mitigate risks.

7.4 Explainability System

The system provides intuitive explanations for each strategy, including its fundamental logic and how signals are triggered. It clarifies why a buy or sell signal appears at specific times and displays the current values and historical context of key technical indicators.

7.5 Frontend Implementation Details

Interactive price charts visualize the performance of backtested strategies. The current strategy and its buy/sell signals are clearly presented, along with actionable advice. A list of recent trade signals is shown, complete with trigger dates, prices, and underlying reasons. Users can choose from multiple preset strategies and adjust parameters in real time to observe how changes affect performance.

7.6 Integration with the Broader System

Strategy analysis results complement those from the sentiment and financial analysis modules. All components communicate via a custom event system to ensure updates are synchronized. The final recommendation engine incorporates these outputs to deliver comprehensive advice. Technical indicator data is also shared across the system for broader analysis.

8: Functional and Logical Breakdown for Final Recommendation

This section provides an in-depth explanation of one of the most critical components of the stock recommendation system: the Final Recommendation module. This module synthesizes insights from previous analysis sources and current market conditions to generate a comprehensive, explainable investment suggestion.

8.1 Overall Workflow

The final recommendation module starts by collecting outputs from the sentiment analysis, financial analysis, and strategy analysis modules. It also gathers current market environment data, including volatility indicators and trend signals. All collected data is sent to the backend API for integrated analysis. The system then employs an OpenAI model to generate a weighted investment score and detailed reasoning. Each factor's contribution is analyzed, and the final recommendation with interpretability insights is presented to the user.

8.2 Data Collection and Integration

Data from the analytical modules include overall sentiment scores, financial metrics like revenue growth and net income growth, and trading strategy outputs such as total returns and signal types. These are stored using local storage and custom event listeners for real-time integration. Additionally, the module retrieves volatility data (e.g., VIX index) and evaluates market trends based on the S&P 500's recent performance. When the primary API fails, fallback sources such as Yahoo Finance are used to ensure data completeness.

8.3 Intelligent Recommendation Generation

The recommendation engine relies on OpenAI's GPT-3.5-turbo model, configured to act as a professional investment advisor. The model runs with a temperature of 0 to ensure consistency and deterministic output. Prompts are designed to yield structured results that include scores, reasoning, and feature weights. The model generates a continuous investment score from -1 (Strong Sell) to 1 (Strong Buy), mapped to corresponding recommendation labels. Market conditions influence factor weights in real time: for example, in high-volatility conditions, strategy analysis gains importance, while in bull or bear markets, weight shifts toward sentiment or financial health accordingly.

8.4 Explainability System

The system dissects the recommendation logic by visualizing the weight of each factor—sentiment, financials, and strategy. Dynamic weight adjustment ensures the sum equals one and reflects real-time conditions. Structured explanations are generated to show how each factor interacts with others and how the broader market landscape influences the final recommendation. A fallback rule engine is used if the AI fails to return valid weights.

8.5 Frontend Implementation Details

The user interface performs data integrity checks to ensure all analytical modules are completed. Visual indicators display the completion status of each module, and incomplete inputs disable the generation function. Recommendation cards use color-coded labels and icons (green for buy, yellow for hold, red for sell) with expandable sections for detailed insights. A dedicated analysis window shows market context, raw model scores, bar charts for factor weights, and explanatory text.

8.6 Integration with the Broader System

This module is tightly integrated with the rest of the system via custom event listeners. It receives analysis results from other modules and resets content dynamically when the user switches the target stock.

Limitations & Future Plans

1. Explainability Model Training and Real-Time Performance:

- The current implementation employs interpretability techniques such as Integrated Gradients and TreeSHAP, which require a large number of evaluation samples to yield robust and reliable attributions. Insufficient sampling may result in less stable or less accurate interpretability outputs.
- Furthermore, achieving real-time performance with these computationally intensive methods is challenging. To maintain low latency, the system may need to deploy high-performance GPUs or adopt an optimized parallel processing infrastructure. Future improvements should consider expanding the evaluation sample set and leveraging advanced hardware acceleration to enhance both robustness and responsiveness.

2. Functional Maturity and Commercialization Readiness:

- At present, the system offers a relatively basic set of functionalities. While this level of service is sufficient for personal or academic use, it does not yet meet the comprehensive feature set required for commercial applications.
- By addressing scalability, integrating additional business-level security and compliance measures, and further refining the user interface, the system could evolve into a commercially viable platform.

3. Trust and Transparency in Model Interpretability:

- Although several advanced interpretability methods have been integrated to enhance transparency, the current explanations remain relatively elementary. They offer preliminary insights into decision-making processes but may not fully dispel the inherent "black box" concerns associated with complex models for all users.

- To further bolster user trust, future developments might incorporate additional layers of human oversight, such as expert auditing or manual review of outputs. Moreover, implementing flexible operational modes, allowing the system to switch between automated explanations and expert-guided interpretations, or to adjust weightings and thresholds for different indicators, could significantly enhance transparency and reliability. While such capabilities are currently aspirational, they represent a promising direction for future refinement.

Conclusion

This project demonstrates a comprehensive approach to stock recommendation by unifying real-time sentiment analysis, fundamental financial evaluation, and technical strategy testing into a single framework. A key highlight of the system is its integration of explainability techniques—such as Integrated Gradients, Self-Attention, and TreeSHAP—which provide users with clear, interpretable insights into the decision-making process. By making the inner workings of each analytical module transparent, the system fosters greater user trust and facilitates informed investment decisions.

While the current implementation serves as a robust proof of concept, it also exposes areas for further enhancement, including the need for increased computational efficiency and expanded functionality for commercial deployment. Overall, the project underscores the practical value of explainable analysis in financial applications, paving the way for future developments that combine in-depth analytics with user-accessible design.