Enhancing Trading Intelligence: A Comprehensive Resource Report for SymbolikAI Development

SymbolikAI, a modular trading intelligence system under development, aims to revolutionize trade signal generation by integrating technical indicators, probabilistic modeling, and explainability techniques. A core objective is to equip the system with reasoning and introspection capabilities, enabling it to analyze past performance, understand the factors contributing to successes and failures, and subsequently refine its decision-making processes. The system's architecture currently includes layers for data storage, signal parsing, backtesting and analytics, and commentary generation, with a future meta-model layer planned for evaluating signal quality and incorporating probabilistic modeling. This report identifies key resources, including GitHub repositories, research papers, and open-source tools, to support the rapid iteration and evolution of SymbolikAI, particularly in the areas of probabilistic modeling, explainability using SHAP, meta-modeling, rule-based systems, and the application of Large Language Models (LLMs) and symbolic AI.

Leveraging Probabilistic and Bayesian Modeling for Trading Decisions

The integration of probabilistic and Bayesian modeling stands as a cornerstone for quantifying uncertainty and making informed trading decisions within SymbolikAI. Several GitHub repositories offer valuable insights and tools for this purpose.

GluonTS, available at awslabs/gluonts [1], is a Python package specifically designed for probabilistic time series modeling, with a strong emphasis on deep learning-based models built on PyTorch and MXNet. This library includes pre-trained models like Chronos, which can generate accurate probabilistic predictions for new time series without requiring prior training on that specific data. The availability of such pre-trained models presents an immediate opportunity for SymbolikAI to experiment with probabilistic forecasting and potentially accelerate the development of its signal generation layer. Given that SymbolikAI's pipeline is Python-based, GluonTS's reliance on PyTorch and MXNet facilitates smoother integration. The library's focus on generating probability distributions over future outcomes, rather than just point estimates, aligns with the goal of quantifying uncertainty in trading predictions.

pystocksim, found at realsurya/pystocksim [2], provides a Python program for simulating and calculating the probability of profit for stock trading. It employs Monte Carlo methods with Geometric Brownian Motion to predict short-term stock price movements. While the underlying model has limitations, such as the assumption of constant drift and volatility, it serves as a practical illustration of how probabilistic modeling can be applied to quantify the likelihood of different trading outcomes. This approach of translating probabilistic thinking into a tangible metric like the probability of profit offers a valuable perspective for SymbolikAI's development, even if more sophisticated models are eventually adopted.

The repository associated with the book **Probabilistic ML for Finance and Investing**, located at dkanungo/Probabilistic-ML-for-finance-and-investing [3], presents a broader perspective on the application of probabilistic machine learning in finance. It emphasizes the ability of these models to handle uncertainty inherent in financial markets and to incorporate existing knowledge into the modeling process. The book's focus on generative ensembles and continuous learning resonates with SymbolikAI's objectives of self-improvement. The code examples within this repository are likely to provide deeper practical understanding of implementing probabilistic ML concepts in a financial context, particularly in handling noisy data and quantifying uncertainty.

Awesome Systematic Trading, a curated list of resources at paperswithbacktest/awesome-systematic-trading [4], includes a wide array of libraries, trading strategies, and educational materials relevant to systematic trading. The sections on Machine Learning and Time Series Analysis within this collection are likely to contain further resources and tools related to probabilistic modeling that could be beneficial for SymbolikAI's development. This repository acts as a central discovery point for a broader ecosystem of quantitative trading tools and techniques.

For a deeper dive into Bayesian modeling techniques, the tutorial **Bayesian Modelling in Python** at markdregan/Bayesian-Modelling-in-Python [5] offers a hands-on guide using the PyMC3 library. It covers fundamental topics such as parameter estimation, model checking, hierarchical modeling, regression, survival analysis, and A/B testing from a Bayesian perspective. The tutorial format, with its practical code examples, provides an excellent avenue for understanding the application of Bayesian methods, which can be adapted to financial time series data within SymbolikAI.

PyBATS, available at lavinei/pybats [6], is a Python package specifically tailored for Bayesian time series modeling and forecasting. It focuses on Dynamic Generalized Linear Models (DGLMs) that can handle various observation distributions, including Normal, Poisson, Bernoulli, and Binomial. This library's specialization in Bayesian time series analysis makes it particularly relevant for SymbolikAI. The ability to model different types of financial data, such as price changes as a Bernoulli distribution, offers a significant advantage for capturing the nuances of market behavior.

PyMC, found at pymc-devs/pymc [7], is a powerful and widely used Python package for Bayesian statistical modeling. It offers advanced Markov chain Monte Carlo (MCMC) and variational inference algorithms, and it leverages PyTensor for computational optimization. The robustness and extensive features of PyMC make it a strong candidate for implementing complex Bayesian models within SymbolikAI. The efficient computation provided by PyTensor is crucial for a trading system that may require real-time analysis.

The GitHub organization **Bayes** at mybayes [8] hosts several repositories related to quantitative trading. While the specific use of Bayesian methods within these repositories is not immediately evident, exploring them could potentially uncover practical implementations of Bayesian trading strategies or related components that could inform SymbolikAI's development.

Finally, the **Machine Learning for Trading** repository at stefan-jansen/machine-learning-for-trading [9] includes a dedicated chapter on Bayesian Machine Learning. This chapter explains how Bayesian statistics can be applied to trading, covering topics like dynamic Sharpe ratios and pairs trading, and demonstrates probabilistic programming using PyMC3. This resource directly addresses the application of Bayesian methods in a trading context, providing specific examples that are highly relevant to SymbolikAI's objectives.

Applying SHAP for Explainable Financial Models

Explainability is a key requirement for SymbolikAI, and SHAP (SHapley Additive exPlanations) offers a powerful framework for understanding the output of machine learning models. Several GitHub repositories demonstrate its application in finance.

The core **SHAP library**, located at shap/shap [10], provides a unified approach to explaining the output of any machine learning model by leveraging Shapley values from game theory. It offers efficient algorithms for tree-based models, deep learning models, and model-agnostic explanations. The library's versatility makes it a fundamental tool for implementing explainability in SymbolikAI, regardless of the specific machine learning models used for signal generation or meta-modeling. Its extensive documentation and examples facilitate integration into Python-based pipelines.

The **Machine Learning for Trading** repository at stefan-jansen/machine-learning-for-trading [12] also discusses the use of SHAP to interpret gradient boosting models, a technique commonly employed in financial modeling. This highlights the direct relevance of SHAP for understanding the feature importance and impact within such models, which SymbolikAI might utilize.

shap-select, found at transferwise/shap-select [13], is a library specifically designed for feature selection in gradient boosting models using regression on feature Shapley values. This offers a practical application of SHAP that could aid in refining the feature set used for SymbolikAI's trading models, potentially improving performance and interpretability by automatically identifying the most relevant factors.

The repository **shap-values** at pablo14/shap-values [14] provides code in R demonstrating how to interpret SHAP values using an XGBoost model. While the code is in R, the underlying principles and visualization techniques for SHAP values are directly transferable to a Python implementation within SymbolikAI. This example can serve as a clear illustration of how to obtain and visualize feature contributions to model predictions.

SHAP-Explainable-Lexicon-Model at hristijanpeshov/SHAP-Explainable-Lexicon-Model [15] showcases an advanced application of SHAP in finance for sentiment analysis. It proposes a methodology using transformers and SHAP to learn financial lexicons, demonstrating how SHAP can be used to understand the contribution of individual words to sentiment predictions. If SymbolikAI incorporates news or other textual data, this repository offers valuable insights into leveraging SHAP for interpreting the impact of textual features.

Finally, **cf-shap** at jpmorganchase/cf-shap [16] is a Python package implementing algorithms for counterfactual feature attribution explanations based on SHAP. This more advanced technique can provide deeper understanding of a model's decision-making process by showing how changes in input features would alter the output. This could enhance the reasoning capabilities of SymbolikAI by allowing for "what-if" analysis of trading scenarios.

Developing Meta-Models and Introspection Layers for Trading Systems

To enhance signal quality and enable self-improvement, SymbolikAI aims to incorporate a meta-model and introspection layer. Several GitHub projects offer relevant architectures and techniques.

The **meta-labeling** repository at hudson-and-thames/meta-labeling [17] provides the codebase for research papers on meta-labeling in finance. It covers the theory behind meta-labeling as a machine learning layer that sits on top of a primary strategy to improve position sizing and filter false positives. It also explores various model architectures for meta-labeling and discusses calibration and ensemble methods. This repository is a crucial resource for understanding and implementing meta-labeling techniques to improve the quality of SymbolikAI's trading signals.

meta-labelling-architecture at LNshuti/meta-labelling-architecture [18] demonstrates a full-stack application that automatically trades Bitcoin using meta-labeling with ensemble modeling on AWS. This provides a practical example of how to integrate meta-labeling into a complete trading system, including data acquisition, model building, backtesting, and cloud deployment. It offers a valuable blueprint for architecting SymbolikAI's meta-model layer.

Meta_Labeling at dreyhsu/Meta_Labeling [19] presents another approach to using meta-labeling, focusing on determining the size of a bet based on the predicted success rate of a trade. This repository provides a specific application of meta-labeling for optimizing position sizing, a critical aspect of risk management that SymbolikAI will need to address.

FinMem-LLM-StockTrading at pipiku915/FinMem-LLM-StockTrading [20] introduces FinMem, an LLM-based agent framework for financial decision-making. While not strictly meta-modeling, its layered memory module designed for interpretability and real-time tuning aligns with the concept of an introspection layer that can learn from past experiences and refine decisions. This framework offers an interesting perspective on building an intelligent trading agent with self-improvement capabilities.

The **Machine Learning for Trading** repository at stefan-jansen/machine-learning-for-trading [21] covers a broad range of ML techniques for trading, which implicitly includes concepts relevant to evaluating and improving signal quality, forming the basis of a meta-model layer. Exploring this repository can provide broader insights into designing and evaluating trading strategies using machine learning.

While **MetaModels** at metamodels [22] appears to be focused on data models for web content management and is likely not relevant to SymbolikAI's needs, the repository **precise** at microprediction/precise [23] contains literature on portfolio optimization and generative models. This might offer insights into advanced techniques for managing and potentially improving trading strategies, although it is not directly focused on meta-models.

Best of Algorithmic Trading at merovinh/best-of-algorithmic-trading [24] is a curated list of algorithmic trading projects. While it does not explicitly mention meta-models or introspection layers in the snippet, it is possible that some of the listed projects incorporate these concepts. Further exploration of this list could reveal relevant implementations or frameworks. Similarly, the topic **trading-algorithms** on GitHub [25] might contain repositories that have implemented introspection layers or meta-modeling techniques, warranting further investigation.

Implementing Commentary Generation and Rule-Based Logic Engines

Generating human-readable commentary for trade signals and implementing rule-based logic for post-trade reflection are crucial for enhancing SymbolikAI's intelligence and explainability. Several GitHub resources provide valuable examples and tools.

TradingBot at Gifted87/TradingBot [26] uses Google's Gemini AI to perform technical analysis and generate trading signals with commentary based on Smart Money Concepts (SMC) and Wyckoff methods. This is a direct example of a system that achieves the goal of generating both trading signals and human-readable explanations using an LLM, making it highly relevant to SymbolikAI's commentary layer development.

Yukina AI at rig-ai-research/yukina [27] is an AI assistant that generates intelligent market commentary and responds to market-related queries using OpenAI's GPT-3.5/4. This project demonstrates the feasibility of using LLMs to create engaging and informative market commentary, offering inspiration for SymbolikAI's commentary generation capabilities.

While **Trading Signal Generation** at quantra-go-algo/Trading_signal_cv.py [28] and **Stock Price Buy/Sell Signal Generator** at r0han01/Stock-Price-Buy-Sell-Signal-Generator [29] focus on generating trading signals, they might provide foundational code that can be integrated with a commentary generation module within SymbolikAI.

For implementing rule-based logic, **Rules Engine** at microsoft/RulesEngine [30] is a library for abstracting business logic and rules out of a system. This general-purpose rules engine could be adapted for SymbolikAI's post-trade reflection, allowing for the definition of specific conditions and actions based on trade outcomes and market behavior.

The description of rules for a model matching engine in a GitHub gist [31] provides detailed examples of how to define precise and logical rules for market interactions. While focused on order execution, the principles outlined can inform the design of rules for analyzing trading behavior and outcomes in SymbolikAI's post-trade reflection module.

Rule-based-forex-trading-system at zzzac/Rule-based-forex-trading-system [32] implements 16 trading rules based on various crossover strategies for Forex trading. Although focused on signal generation, it offers a concrete example of how to define and implement rule-based trading strategies using technical indicators, which could be adapted for post-trade analysis in SymbolikAI.

Exploring the topic **rules-engine** on GitHub [33] can lead to a wider range of potential libraries and examples for building the post-trade reflection module in SymbolikAI.

ZEN Business Rules Engine at gorules/zen [34] is an open-source, cross-platform rules engine written in Rust with bindings for Python. It allows for the loading and execution of JSON Decision Models (JDM), offering a powerful and flexible option for implementing complex rule-based logic within SymbolikAI. The use of JSON for defining rules facilitates easy management and modification.

Investigating Symbolic AI and Case-Based Reasoning in Financial Markets

Symbolic AI and case-based reasoning (CBR) offer alternative approaches to enhancing the intelligence of trading systems. Several GitHub resources explore these paradigms in finance.

Curated lists like **Awesome AI in Finance** at ihobbang250/Awesome-AI-in-Finance and georgezouq/awesome-ai-in-finance [35] include research papers and resources on AI in finance, with sections on reasoning and trading agents that may involve symbolic AI or CBR techniques. These lists serve as valuable discovery points for relevant research and projects.

FinRobot at AI4Finance-Foundation/FinRobot [37] is an open-source AI agent platform for financial analysis using LLMs. While primarily focused on LLMs, its modular design and emphasis on financial reasoning could potentially incorporate elements of symbolic AI or CBR in the future, or inspire such integrations within SymbolikAI.

Financial Market Data Analysis at radoslawkrolikowski/financial-market-data-analysis [38] focuses on deep learning for stock price prediction but highlights the complexity of financial data and feature engineering, which are relevant to any AI-driven trading system, including those using symbolic AI or CBR. Similarly, **AI in Finance Market Prediction** at llSourcell/AI_in_Finance [39] provides a general overview of AI applications in finance, offering context for exploring different AI techniques.

Several repositories focus specifically on case-based reasoning. **case-based-reasoning** at sipemu/case-based-reasoning [40] provides an R interface for CBR using machine learning methods. While in R, it explains the core principles of CBR and its potential application in observational studies and with statistical models. **Case-Based** at Case-Based/casebased [41] is a Python library implementing CBR principles, aiming to make CBR a more popular method for solving issues using machine learning. **cbrs** at yammadev/cbrs [43] offers a Python-based CBR system, modeling reasoning as primarily memory-based and providing a four-step process of retrieve, reuse, revise, and retain. Finally, **CBRkit** at wi2trier/cbrkit [44] is a customizable and modular toolkit for CBR in Python, offering tools for loading cases, defining similarity measures, and retrieving relevant cases. The availability of these CBR libraries in Python makes it feasible for SymbolikAI to explore this paradigm for building its reasoning and introspection layers.

The Role of Large Language Models in Financial Reasoning

Large Language Models (LLMs) hold significant potential for enhancing the reasoning and commentary generation capabilities of SymbolikAI. The curated lists of research papers in **Awesome AI in Finance** [35] indicate a strong and growing interest in applying LLMs to various financial tasks, including reasoning. Research on the ability of LLMs like ChatGPT to forecast stock prices [35] demonstrates the direct relevance of this technology to financial prediction, although results may vary. The practical application of LLMs in **FinRobot** [37] for tasks like report writing and potentially trading strategy generation provides a model for how SymbolikAI can integrate LLMs into its broader architecture.

Meta-Reasoning and Meta-Learning for Trading Strategy Refinement

To enable SymbolikAI to learn from its past performance and refine its trading strategies, the concepts of meta-reasoning and meta-learning are crucial. The **FinMem** framework [20] with its ability to self-evolve its knowledge and continuously refine trading decisions aligns with the principles of meta-learning. The mention of **QuantAgent** [35] aiming for self-improvement in trading using LLMs further highlights the potential of these techniques for building a self-improving trading system like SymbolikAI. FinMem's focus on interpretability and its memory module could provide a mechanism for SymbolikAI to analyze past trade failures and understand the underlying reasons, directly addressing the user's initial query.

Architectural Considerations for Explainable and Adaptive Trading Engines

Designing a robust and flexible architecture is essential for integrating the various components of SymbolikAI. The **meta-labelling-architecture** repository [18] provides a concrete example of a trading system architecture that incorporates meta-labeling, offering a potential starting point for SymbolikAI's overall design. **FinRobot's** layered architecture for financial AI agents [37] presents a broader perspective on structuring a complex system with multiple capabilities, ensuring effective interaction between different components like data, signals, commentary, and the meta-model. A key architectural consideration for SymbolikAI will be to define clear interfaces and communication protocols between its different layers to ensure seamless integration and information flow.

Post-Processing Trading Signals: Filtering, Commentary, and Reinforcement

Post-processing trading signals is crucial for enhancing their quality and utility. The resources on meta-labeling [17] directly address the filtering of trading signals by using a secondary model to assess their quality. Implementing meta-labeling in SymbolikAI can significantly improve the reliability of its generated signals. The resources on commentary generation [26] demonstrate how LLMs can be used to provide human-readable explanations for trading signals, enhancing SymbolikAI's explainability. While not explicitly requested, the concept of reinforcement learning for trading [35] suggests a potential future direction for SymbolikAI to further enhance its adaptive capabilities by learning optimal signal filtering or post-processing strategies based on backtesting results. A comprehensive approach combining probabilistic modeling, explainability analysis using SHAP, rule-based filtering, and meta-labeling is likely to be the most effective strategy for refining trade signals within SymbolikAI.

Conclusion: Integrating Advanced Techniques for a Self-Improving Trading System

The development of SymbolikAI, with its ambitious goals of generating trade signals accompanied by human-readable reasoning and the ability to self-improve, necessitates the integration of several advanced techniques. The exploration of GitHub repositories, research papers, and open-source tools has revealed a rich landscape of resources that can significantly accelerate this development. Probabilistic and Bayesian modeling offers the means to quantify uncertainty, while SHAP provides a robust framework for explainability. Meta-modeling and introspection layers, supported by techniques like meta-labeling, are crucial for enhancing signal quality and enabling learning from past performance. LLMs present a powerful tool for generating insightful commentary, and symbolic AI and case-based reasoning offer alternative paradigms for building intelligent systems.

The identified resources highlight the importance of a modular architecture for SymbolikAI, allowing for the effective integration of these diverse components. As the development progresses, a focus on clearly defined interfaces and communication protocols between the data, signal, backtesting, commentary, and meta-model layers will be essential. The iterative nature of building such a complex system should be embraced, with continuous learning and refinement based on both successes and failures.

The following table summarizes some of the most promising GitHub repositories for the continued development of SymbolikAI:

Repository Name	**GitHub URL**	**Key Features/Relevance to SymbolikAI**	**Potential Application in SymbolikAI**
GluonTS	https://github.com/awslabs/gluonts	Probabilistic time series modeling, deep learning models, pre-trained models for forecasting.	Initial experimentation with probabilistic forecasting for signal generation.
shap/shap	https://github.com/sh	Unified approach to explain ML model	Implementing explainability for

	ap/shap	outputs, efficient for tree-based and deep learning models.	trade signals generated by various models.
hudson-and-thames/meta-labeling	https://github.com/hudson-and-thames/meta-labeling	Codebase for research on meta-labeling in finance, covering theory and implementation.	Developing the meta-model layer for improving signal quality and position sizing.
LNshuti/meta-labelling-architecture	https://github.com/LNshuti/meta-labelling-architecture	Full-stack example of automated Bitcoin trading using meta-labeling on AWS.	Architectural inspiration for integrating meta-labeling into a complete trading system.
Gifted87/TradingBot	https://github.com/Gifted87/TradingBot	Uses Gemini AI for technical analysis and generates trading signals with commentary.	Studying the implementation of AI-powered commentary generation for trade signals.
rig-ai-research/yukina	https://github.com/rig-ai-research/yukina	AI assistant generating market commentary using OpenAI's GPT-3.5/4.	Exploring the use of LLMs for generating insightful market commentary.
gorules/zen	https://github.com/gorules/zen	Cross-platform Business Rules Engine with Python bindings, uses JSON Decision Models.	Implementing rule-based logic for post-trade reflection and signal filtering.
Case-Based/casebased	https://github.com/Case-Based/casebased	Python library implementing Case-Based Reasoning principles.	Experimenting with CBR for building the reasoning and introspection layers of SymbolikAI.
wi2trier/cbrkit	https://github.com/wi	Customizable and modular toolkit for	Exploring advanced CBR techniques for

	2trier/cbrkit	Case-Based Reasoning in Python.	enhancing SymbolikAI's reasoning capabilities.
stefan-jansen/machine-learning-for-trading	https://github.com/stefan-jansen/machine-learning-for-trading	Covers various ML techniques for trading, including Bayesian methods and SHAP.	Provides practical examples and context for applying Bayesian ML and SHAP in a trading environment.

Works cited

1. awslabs/gluonts: Probabilistic time series modeling in Python - GitHub, accessed March 30, 2025, https://github.com/awslabs/gluonts
2. realsurya/pystocksim: Python program for simulating and calculating probability of profit for stock trading. Simulations are done though the Monte-Carlo Method with Geometric Brownian Motion to predict future closing price of a stock in the short-term. - GitHub, accessed March 30, 2025, https://github.com/realsurya/pystocksim
3. dkanungo/Probabilistic-ML-for-finance-and-investing: Probabilistic Machine Learning for Finance and Investing: A Primer to Generative AI with Python - GitHub, accessed March 30, 2025, https://github.com/dkanungo/Probabilistic-ML-for-finance-and-investing
4. A curated list of awesome libraries, packages, strategies, books, blogs, tutorials for systematic trading. - GitHub, accessed March 30, 2025, https://github.com/paperswithbacktest/awesome-systematic-trading
5. markdregan/Bayesian-Modelling-in-Python: A python ... - GitHub, accessed March 30, 2025, https://github.com/markdregan/Bayesian-Modelling-in-Python
6. lavinei/pybats: Bayesian time series forecasting and ... - GitHub, accessed March 30, 2025, https://github.com/lavinei/pybats
7. pymc-devs/pymc: Bayesian Modeling and Probabilistic Programming in Python - GitHub, accessed March 30, 2025, https://github.com/pymc-devs/pymc
8. Bayes - GitHub, accessed March 30, 2025, https://github.com/mybayes
9. Bayesian ML: From recession forecasts to dynamic pairs trading - GitHub Pages, accessed March 30, 2025, https://stefan-jansen.github.io/machine-learning-for-trading/10_bayesian_machine_learning/
10. shap/shap: A game theoretic approach to explain the output ... - GitHub, accessed March 30, 2025, https://github.com/shap/shap
11. shaoshanglqy/shap-shapley - GitHub, accessed March 30, 2025, https://github.com/shaoshanglqy/shap-shapley
12. Boosting your Trading Strategy | Machine Learning for Trading - GitHub Pages, accessed March 30, 2025, https://stefan-jansen.github.io/machine-learning-for-trading/12_gradient_boostin

g_machines/
13. transferwise/shap-select: A library for feature selection for ... - GitHub, accessed March 30, 2025, https://github.com/transferwise/shap-select
14. pablo14/shap-values: Shap values for model interpretation - GitHub, accessed March 30, 2025, https://github.com/pablo14/shap-values
15. hristijanpeshov/SHAP-Explainable-Lexicon-Model: This project proposes a novel methodology to automatically learn financial lexicons that outperform the benchmark Loughran-McDonald lexicon in sentiment analysis tasks - GitHub, accessed March 30, 2025, https://github.com/hristijanpeshov/SHAP-Explainable-Lexicon-Model
16. jpmorganchase/cf-shap: Counterfactual SHAP: a framework for counterfactual feature importance - GitHub, accessed March 30, 2025, https://github.com/jpmorganchase/cf-shap
17. hudson-and-thames/meta-labeling: Code base for the meta ... - GitHub, accessed March 30, 2025, https://github.com/hudson-and-thames/meta-labeling
18. LNshuti/meta-labelling-architecture: Automated Trading ... - GitHub, accessed March 30, 2025, https://github.com/LNshuti/meta-labelling-architecture
19. dreyhsu/Meta_Labeling: Meta labeling is a method of determining the size of the bet. - GitHub, accessed March 30, 2025, https://github.com/dreyhsu/Meta_Labeling
20. pipiku915/FinMem-LLM-StockTrading: FinMem: A Performance-Enhanced LLM Trading Agent with Layered Memory and Character Design - GitHub, accessed March 30, 2025, https://github.com/pipiku915/FinMem-LLM-StockTrading
21. stefan-jansen/machine-learning-for-trading: Code for ... - GitHub, accessed March 30, 2025, https://github.com/stefan-jansen/machine-learning-for-trading
22. MetaModels - GitHub, accessed March 30, 2025, https://github.com/metamodels
23. precise/LITERATURE.md at main · microprediction/precise - GitHub, accessed March 30, 2025, https://github.com/microprediction/precise/blob/main/LITERATURE.md
24. merovinh/best-of-algorithmic-trading: A ranked list of algorithmic trading open-source libraries, frameworks, bots, tools, books, communities, education materials. Updated weekly. - GitHub, accessed March 30, 2025, https://github.com/merovinh/best-of-algorithmic-trading
25. trading-algorithms · GitHub Topics, accessed March 30, 2025, https://github.com/topics/trading-algorithms
26. Gifted87/TradingBot: A multi AI Agents program that ... - GitHub, accessed March 30, 2025, https://github.com/Gifted87/TradingBot
27. rig-ai-research/yukina - GitHub, accessed March 30, 2025, https://github.com/rig-ai-research/yukina
28. Trading Signal Generation - GitHub Gist, accessed March 30, 2025, https://gist.github.com/quantra-go-algo/c8b30cca1159189582c93d92e87f3b45
29. r0han01/stock-price-buy-sell-signal-generator - GitHub, accessed March 30, 2025, https://github.com/r0han01/Stock-Price-Buy-Sell-Signal-Generator
30. microsoft/RulesEngine: A Json based Rules Engine with extensive Dynamic expression support - GitHub, accessed March 30, 2025,

https://github.com/microsoft/RulesEngine
31. Market Order Matching Engine - GitHub Gist, accessed March 30, 2025, https://gist.github.com/jdrew1303/e06361070468f6614d52216fb91b79e5
32. zzzac/Rule-based-forex-trading-system - GitHub, accessed March 30, 2025, https://github.com/zzzac/Rule-based-forex-trading-system
33. rules-engine · GitHub Topics, accessed March 30, 2025, https://github.com/topics/rules-engine
34. gorules/zen: Open-source Business Rules Engine for your ... - GitHub, accessed March 30, 2025, https://github.com/gorules/zen
35. ihobbang250/Awesome-AI-in-Finance - GitHub, accessed March 30, 2025, https://github.com/ihobbang250/Awesome-AI-in-Finance
36. georgezouq/awesome-ai-in-finance: A curated list of awesome LLMs & deep learning strategies & tools in financial market. - GitHub, accessed March 30, 2025, https://github.com/georgezouq/awesome-ai-in-finance
37. FinRobot: An Open-Source AI Agent Platform for Financial Analysis using LLMs - GitHub, accessed March 30, 2025, https://github.com/AI4Finance-Foundation/FinRobot
38. radoslawkrolikowski/financial-market-data-analysis: Real-Time Financial Market Data Processing and Prediction application - GitHub, accessed March 30, 2025, https://github.com/radoslawkrolikowski/financial-market-data-analysis
39. AI_in_Finance/Market Prediction.ipynb at master - GitHub, accessed March 30, 2025, https://github.com/llSourcell/AI_in_Finance/blob/master/Market%20Prediction.ipynb
40. sipemu/case-based-reasoning - GitHub, accessed March 30, 2025, https://github.com/sipemu/case-based-reasoning
41. Case Based Reasoning - GitHub, accessed March 30, 2025, https://github.com/Case-Based
42. accessed December 31, 1969, https://github.com/Case-Based/casebased
43. yammadev/cbrs: Case-based Reasoning (CBR) System - GitHub, accessed March 30, 2025, https://github.com/yammadev/cbrs
44. wi2trier/cbrkit: Customizable Case-Based Reasoning (CBR ... - GitHub, accessed March 30, 2025, https://github.com/wi2trier/cbrkit