

Research: Liquidity Model, Portfolio Optimization

Aryan Jhaveri¹ and Sang Park²

[†]These authors contributed equally to this work.

*The Research is baesd on DeepMM Platform

Abstract

IN THIS PAPER it will be shown that, “the disordered trades” can be optimized through DeepMM’s trading platform with high Portfolio Optimization. If DeepMM’s internal technology can capture the market better through liquidity models, then we expect the profit to be higher through the intra snapshots of trades for portfolio optimization albeit market regime changes. To design alpha strategy, we make two perpendicular planes of X-axis and Y-axis such that moving $t + \tau$ adapts to the relation to the market and provide using Jaccard Similarity along with liquidity models to provide automated trades that are grounded based on MILP (Linear Algebra) computation.

Keywords: Multifactor Ridge Model; Mixed Linear Algebra; MRR liquidity model, Threshold Design, Jaccard Similarity, Alpha Signal, Genetic Algorithm/PSO Portfolio Tuning

This DeepMM Research paper begins with background about automated trading. Traders will prefer to use the electronic system as such transaction does not involve human loops that can cost fees and time. Such automation provides lower marginal commissions and imbalances in liquidity are less likely for liquidity are less likely for electronic brokers to execute, as there are services for manual payments. Also, the automated trading becomes useful for time periods when there is high volume of securities traded at within a short period of time. In order to incorporate automation, we find methodologies to find alpha signal that can be traded on its own with safe volatility level. In terms of methodology for empirical study, We design a threshold using Jaccard similarity and startup’s liquidity computation models and simulate trade particle particles in parallelepiped regions and evaluate the performance that indicates that DeepMM’s technology proves excellence in sharp ratio and return.

Jaccard Similarity as selection of portfolio stock: The Jaccard Similarity lies in the concept that In a time interval of t , the x =coordinate indicates the Jaccard similarity divergence while the y =coordinate indicate other. In a time interval τ , the X -coordinates of the individual particles will increase by τ , where has a different (positive or negative) value for each particle. A certain probability distribution law will What is Jaccard Similarity? Jaccard similarity is used to to measure the temporal stability of selected corporate-bond portfolios by comparing the overlap of bond sets across adjacent or rolling periods. Higher Jaccard values indicate more persistent portfolio membership and lower implied turnover. The measure was incorporated into the execution framework as a stability filter, helping distinguish persistent signals suitable for automation from unstable opportunities requiring greater manual oversight.

1.1: Liquidity Model Types

L_{TRACE_MRR} : Madhavan–Richardson–Roomans Liquidity Measure

The Madhavan–Richardson–Roomans (MRR) model is used to characterize transaction costs and short-run price changes as a function of trade direction. In this study, the MRR framework provides a market-microstructure-based measure of corporate bond liquidity using observed TRACE transaction prices and inferred trade signs.

The model captures how market prices evolve over time in response to both trade-related information and unanticipated public news. Under the MRR framework, the efficient price follows a martingale process, which allows the model to distinguish information-driven price changes from temporary liquidity-related deviations from fair value. This is particularly useful for identifying the component of observed price movements associated with unexpected order flow and market-making frictions.

$$R_1 = E [x_t(P_{t+1} - P_t)], \quad (1)$$

where x_t denotes the direction of trade at time t and P_t denotes the corresponding transaction price. We also define the first-order trade-sign autocorrelation as

$$C_1 = E [x_t x_{t+1}]. \quad (2)$$

The implied MRR liquidity spread is then calculated as

$$L_{TRACE_MRR} = \frac{2|R_1|}{1 - C_1}. \quad (3)$$

L_{AXOR} : Liquidity Measure Based on the AXOR Internal Pricing Model

L_{AXOR} is constructed from DeepMM’s probabilistic pricing framework, AXOR fair value. Unlike liquidity measures based directly on observed transactions or quoted bid–offer spreads,

AXOR generates a distribution of predicted fair values for each bond and date rather than a single point estimate.

For each bond, the AXOR pricing distribution is summarized using percentile estimates ranging from the 5th percentile to the 95th percentile. The AXOR liquidity measure is defined as

$$L_{AXOR,t} = P_{AXOR,t}^{95} - P_{AXOR,t}^{05} \quad (4)$$

where $P_{AXOR,t}^{95}$ and $P_{AXOR,t}^{05}$ denote the 95th- and 5th-percentile AXOR fair-value estimates for a given bond at time t , respectively.

The resulting percentile range captures the degree of uncertainty in the model-implied fair value. A wider L_{AXOR} indicates greater dispersion in the predicted price distribution and is therefore interpreted as weaker liquidity and greater uncertainty regarding the bond's fair value. Conversely, a narrower L_{AXOR} indicates greater agreement within the AXOR pricing distribution and is interpreted as stronger liquidity.

Because the measure is generated directly from the AXOR pricing distribution, L_{AXOR} provides a model-based complement to the transaction-based L_{TRACE_MRR} measure and the quote-based L_{BCQ} measure. This allows the empirical analysis to compare liquidity information obtained from realized transactions, dealer quotations, and probabilistic fair-value estimation.

2.1: Constructing Threshold design for automation

Another round of empirical testing examines whether DeepMM's liquidity model improves the stability of Jaccard similarity. As shown in the results, Jaccard similarity stability improves after incorporating AXOR liquidity into the price-divergence stability model.

The quantity $P_{TRACE} - P_{AXOR}$ captures price divergence between the observed TRACE transaction price and the AXOR estimated fair value. After incorporating DeepMM's liquidity measure, the stability of the daily bond-selection set, as measured by Jaccard similarity, improves.

Threshold Design improves liquidity performance by implementing the DeepMM's Liquidity model AXOR

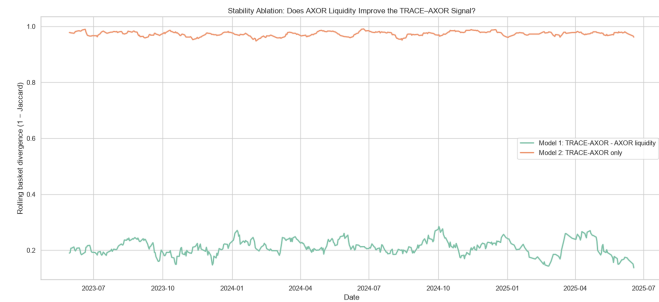


Figure 1 AXOR liquidity model implementation. Result: When we initially used Model 1 with the divergence-capturing model, we noticed that the rolling basket divergence, which captures the day-to-day stability of Jaccard similarity, was very high. Incorporating the AXOR liquidity model substantially improves the stability of the selected liquidity basket. Compared with the TRACE-AXOR signal, which exhibits divergence above 0.9, the AXOR-liquidity specification produces more persistent selections, with divergence around 0.2. A larger version of the figure is provided at the end of the paper.

2.2: Design Threshold for X-axis and Y-axis

We design our threshold such that x-axis is price divergence (TRACE - AXOR) - AXOR liquidity model. And the percentile ranking is recalculated every month with a rule that if trades are over 67th percentile, then we see signals are safe for automation in x-axis quadrant. In terms of Y-axis, we ask whether the trading environment is stable in terms of Jaccard Similarity. If it the divergence is lower than 33rd percentile, then we see the trades to be safe to automate. Then, we have upper right bottom region that the trade simulations are designed to be automated.

Decision for Automation and Human Review

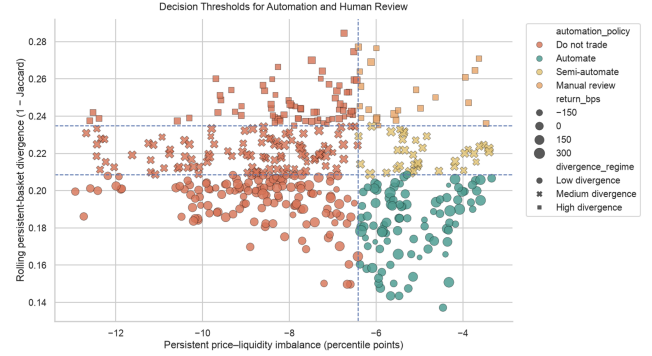


Figure 2 Decision Threshold for Automation and Human Review

Model Specific cut-off analysis

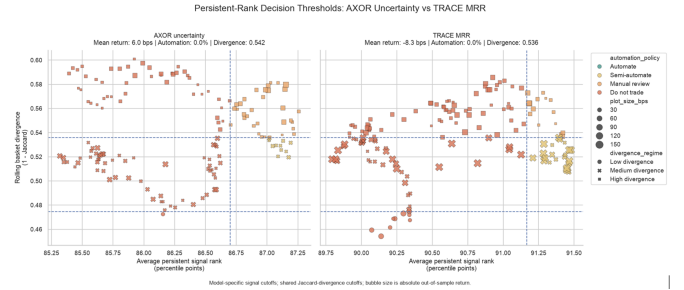


Figure 3 Decision Threshold for Automation and Human Review (Larger Image is available at the end of the research)

A larger version of the figure is provided at the end of the paper.

Result: After simulating 4006k trades, we have around 300k automated trades are within the thresholds as automated. And we also see that compared to Model 0, Model 1 has 2 sharp ratio indicating that AXOR model wins in automated trading for sharp ratio while the TRACE MRR has negative in sharp ratio

Applying MILP Step 1: Multifactor Ridge

After the threshold selects the automtaed trades, then Ridge Machine Learning predicts alpha signal AXOR relative value, BondCliQ relative value, Momentum, and liquidity models of AXOR and Trace to re-learn coefficients from past trading outcomes with these coefficients and come up with the following equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \varepsilon \quad (5)$$

and the y for each of the bond becomes the predictive score to help the policy decide which bonds to rank in stock portfolio. The Ridge Function is based on linear regression with extra penalty on large coefficients. With millions of iterations in historical scenarios, then it uses vectorized model such that Ridge estimate the coefficient vectors into one coefficient to come up with a single fitted model that now simulates predictive score for each of the bonds.

Applying MILP Step 2: Portfolio Construction

The MILP takes Ridge's $\hat{\mu}_i$ and decides the actual weights. Conceptually, MILP explicitly rewards the Ridge expected return, while simultaneously penalizing tracking error, illiquidity, number of names, number of trade tickets, and transaction costs. It also implements position caps, tracks turnover relative to existing holdings, and limits holdings.

Skewness Control

For the Ridge target, data is designed such that only returns from 1 to 99 percentile are kept. For Ridge function, the equation is $(y_i - \hat{y}_i)^2$ and large error can contribute to impact the coefficients heavily impacting the result due to the square.

The second skewness control is that there is a separate transaction-cost tail-control strategy. The design examines the upper quantiles of raw transaction costs and creates capped cost scenarios. Winsorization and transaction-cost tail controls are implemented reduce the influence of extreme observations and highly right-skewed implementation-cost realizations. Specifically, the policy looks at both Ridge predicted return and estimated transaction cost to decide whether to trade or not before as a policy filter before moving into the MILP.

MILP Mixed-Integer Linear Programming

The Mixed-Integer Linear Programming decides "Among eligible trades and existing holdings, which positions should we hold, and at what weights, given the portfolio constraints?"

The MILP represents portfolio construction using continuous allocation weights and binary selection indicators. The vector $\mathbf{w}$ determines the allocation to each directional bond position, while binary variables y_i and z_i indicate position selection and trading activity, respectively. Predicted returns enter the optimizer as a fixed vector $\hat{\mu}$, so expected portfolio return is expressed as the dot product $\mathbf{w}^T \hat{\mu} = \sum_i w_i \hat{\mu}_i$. This expression is linear in the allocation weights. The objective rewards predicted portfolio return while penalizing deviations from target allocations, illiquidity, trading costs, and the number of positions and trading tickets. Position weights and trading amounts are each decomposed into three bounded segments with increasing penalty coefficients, producing piecewise-linear penalties for larger illiquid positions and larger trades.

Auxiliary variables represent absolute deviations from target weights and absolute changes from previous holdings, with each absolute-value relationship expressed through two linear inequalities. These relationships and the portfolio restrictions are assembled into a matrix system, $\mathbf{b}_L \leq \mathbf{A}\mathbf{v} \leq \mathbf{b}_U$, where $\mathbf{v}$ contains the continuous, binary, and auxiliary variables. Linking constraints, $w_i \leq \bar{w}_i y_i$ and $u_i \leq M_i z_i$, connect allocations and trading amounts to the corresponding binary decisions. Additional constraints limit issuer exposure, individual position sizes, total holdings, and trading tickets. The budget constraint, $\sum_i w_i \leq 1$, allows unallocated capital to remain in cash. The

solver therefore determines both portfolio selection and allocation, balancing predicted returns against implementation penalties and the adjustments required from existing holdings.

Ridge Predictive model

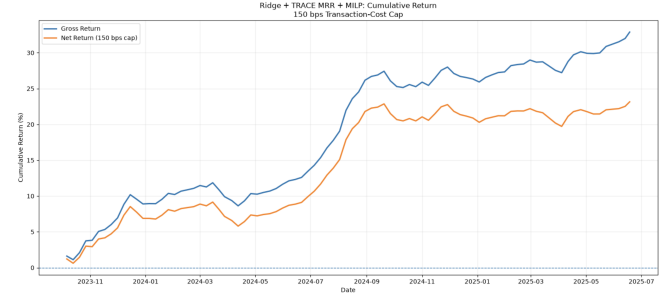


Figure 4 Ridge predictive model using MILP for the holistic trades consisting of both manual and automated that gives over 20 percent return considering the transaction cost

Genetic Algorithm/PSO for Portfolio Tuning

GA/PSO layer is used to find portfolio-design parameters that produce a high net Sharpe ratio before MILP constructs the portfolio.

The algorithm creates a "population" of hundreds of different portfolio models, treating the constraints of each model as hyperparameters.

Then the algorithm evaluates and forms the best boundaries. The algorithm tests how each constraint set handles real-world market friction. The algorithm discovers that if the cardinality constraint is too tight (such as too few bonds), the portfolio lacks diversification and the risk spikes. Under the condition the cardinality constraint is too loose (such as too many bonds), the MRR liquidity signal gets diluted, transaction cost boosts, and the net Sharpe ratio drops.

The algorithm then proceeds to the selection and mutation process. The algorithm eliminates boundaries that result in loss, takes boundaries that result in healthy Sharpe ratio and adjusts or mutates them slightly. These constraints are then added to the next generation of portfolios built.

The algorithm also adapts to the MRR Liquidity Regime. During market panics, liquidity vanishes as transactional activity subsides, and MRR-derived spreads expand significantly. The constraints set depend dynamically on the shifts. If it is a high-liquidity regime, the algorithm detects low execution costs and sets tighter cardinality constraints and higher weight caps to aggressively capture momentum. If it is a low liquidity regime, the algorithm automatically adapts by expanding the constraints to spread risk and protect the portfolio from getting trapped in illiquid positions. The Figure 7 Below the research indicates the sharp ratio ranging from 1.5 to 1.6 percent.

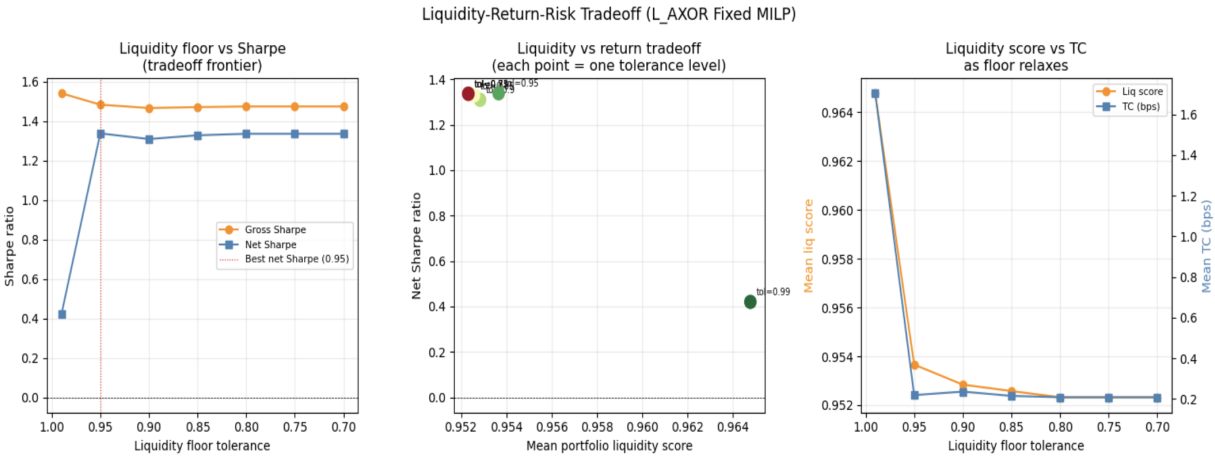


Figure 5 Genetic Algorithm/PSO: Sharp Ratio comparison

Discussion

DeepMM’s internal Liquidity Model has proven to perform better than sophisticated models like MRR used in Microstructure World. Specifically, the automated alpha signal achieved an 8 percent Sharpe Ratio with a downside deviation of 6 percent, while the LCQ Benchmark has a 15 percent downside deviation and posted these 22 weeks. Importantly, such Alpha signal can be automated without human loop since it takes less losses as the algorithm decide selective bonds based on strategies of MILP, Ridge Modeling, GA/PSO Tuning layer to Threshold of Jaccard Similarity and Internal DeepMM’s Liquidity Model. Also, when considering as the whole simulation, then we have over 25 percent in terms of Gross Return as well.

Costa, Luciano da Fontoura. “Further Generalizations of the Jaccard Index.” *arXiv*, 18 Nov. 2021, arXiv:2110.09619v3.

Diether, Karl B., Christopher J. Malloy, and Anna Scherbina. “Differences of Opinion and the Cross Section of Stock Returns.” *The Journal of Finance*, vol. 57, no. 5, Oct. 2002, pp. 2113–2141.

Goyenko, Ruslan Y., Craig W. Holden, and Charles A. Trzcinka. “Do Liquidity Measures Measure Liquidity?” *Journal of Financial Economics*, vol. 92, no. 2, May 2009, pp. 153–181.

Hasbrouck, Joel. “One Security, Many Markets: Determining the Contributions to Price Discovery.” *The Journal of Finance*, vol. 50, no. 4, Sept. 1995, pp. 1175–1199.

Holden, Craig W., Stacey Jacobsen, and Avaniidhar Subrahmanyam. “The Empirical Analysis of Liquidity.” *Foundations and Trends in Finance*, vol. 8, no. 4, 2014, pp. 263–365.

“What Is the Genetic Algorithm?” *MATLAB Help Center*, MathWorks, <https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html>. Accessed 4 Sept. 2026.

Bertsimas, Dimitris, Christopher Darnell, and Robert Soucy. “Portfolio Construction Through Mixed-Integer Programming at Grantham, Mayo, Van Otterloo and Company.” *Interfaces*, vol. 29, no. 1, Jan.–Feb. 1999, pp. 49–66. <https://www.mit.edu/~dbertsim/papers/Finance/Portfolio%20construction%20through%20mixed%20integer%20programming.pdf>. Accessed 5 Sept. 2026.

Costa, Luciano da Fontoura. “Further Generalizations of the Jaccard Index.” *arXiv*, 18 Nov. 2021, arXiv:2110.09619v3, <https://arxiv.org/abs/2110.09619>.

Github Link

Please find the below link for the github: <https://github.com/sup21072/DeepMM>.
booktabs

References



Figure 6 The bottom right quadrant indicate the automated trades where we see high distribution of Positive PnL distribution

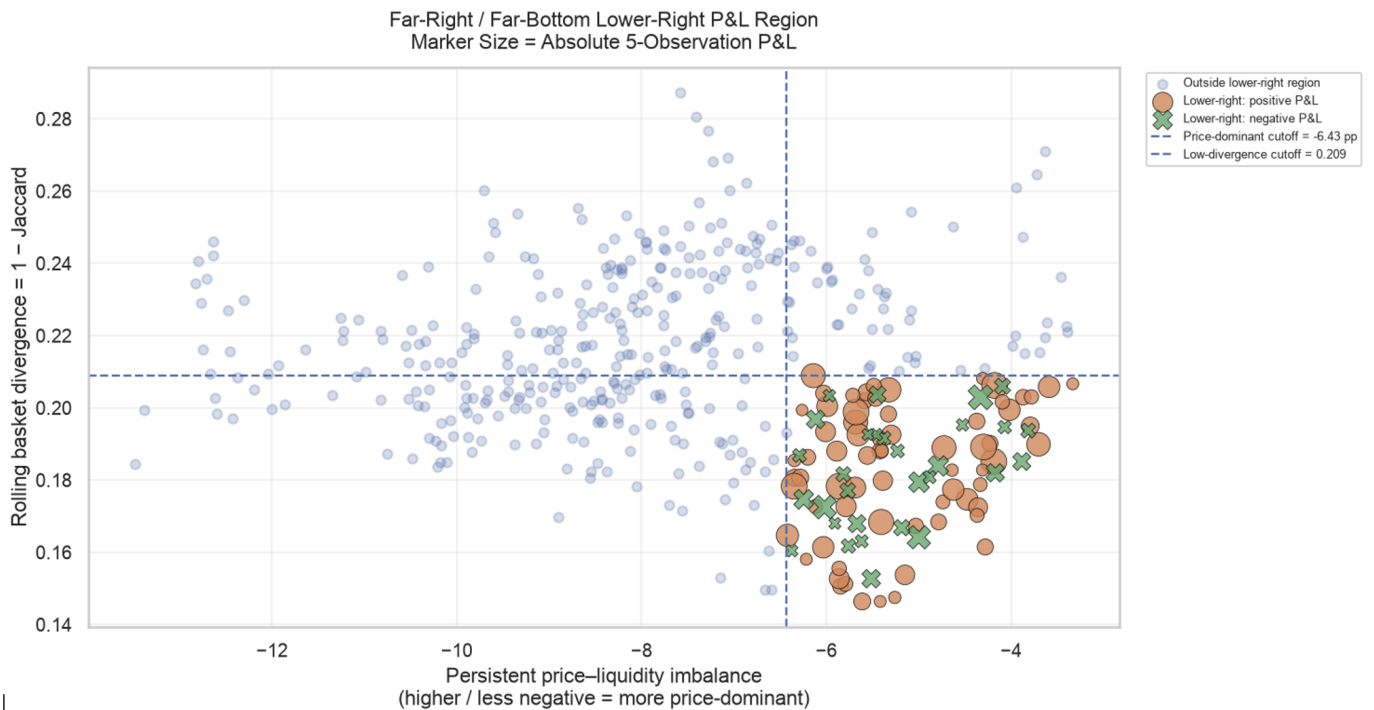


Figure 7 The bottom right quadrant indicate the automated trades where we see high distribution of Positive PnL distributio in response to Liquidity Model MRR

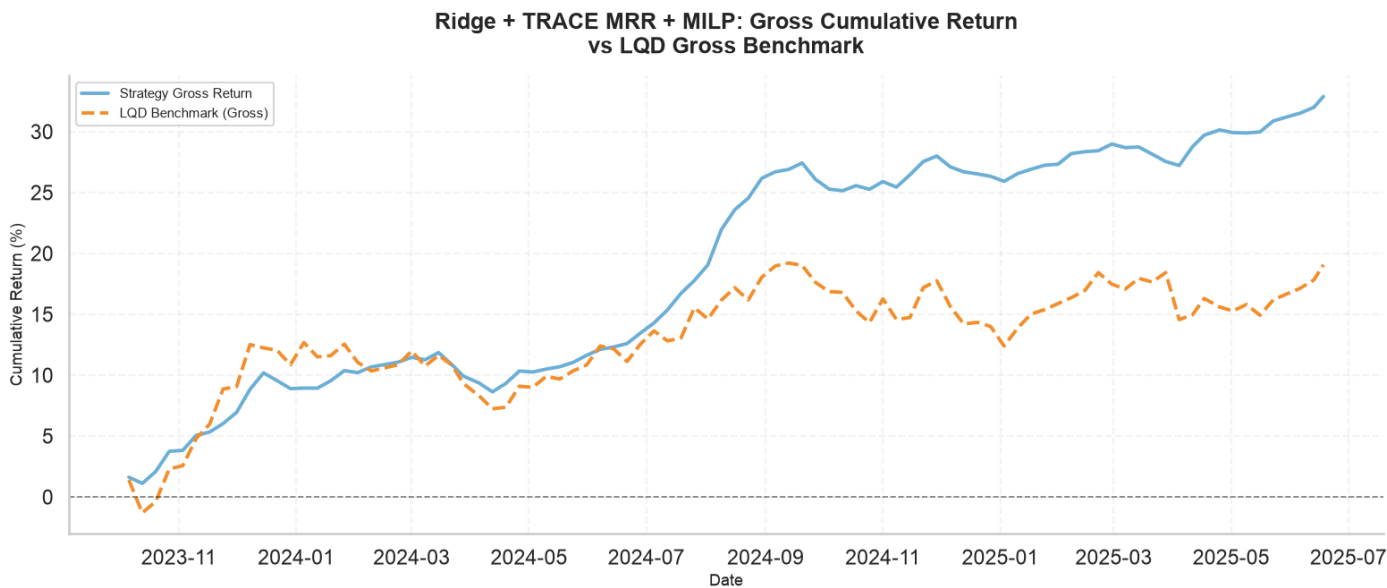


Figure 8 The bottom right quadrant indicate the automated trades where we see high distribution of Positive PnL distributio in response to Liquidity Model MRR

Table 1 Liquidity Model Performance Comparison for 22 weekly observable timeline when Model was active

Model	Sharp Ratio	Sortino	Downside Deviation
DeepMM AXOR Liquidity Model (automated-Threshold)	8.173%	56.498	0.06%
LQD Benchmark Model	-0.419%	-0.552	0.15%

Table 2 Monthly Walk-Forward Trade-Level Automation Results

Model	OOS Trade Points	Automated Trades	Automated Dates	Automated Bonds	Automation Rate	Mean Return (bps)	Median Return (bps)	Hit Rate
Price Divergence–AXOR Liquidity	4,356,778	346,792	98	16,719	0.0796	8.3687	5.8393	0.5661
Price Divergence–TRACE MRR Liquidity	4,356,696	417,784	118	16,185	0.0959	−0.1535	−0.2249	0.4983

Table 3 Divergence and Automation Sharpe-Proxy Metrics

Model	Mean Divergence	Divergence Volatility	95th-Percentile Divergence	False Automation Rate	Automation Sharpe Proxy
Price Divergence–AXOR Liquidity	0.2018	0.0053	0.2091	0.0612	2.1482
Price Divergence–TRACE MRR Liquidity	0.1983	0.0090	0.2087	0.0932	−0.0888

Table 4 Strategy versus LQD Benchmark: Risk and Return Metrics

Strategy	Cumulative Return	Annualized Return	Annualized Volatility	Sharpe Ratio	Downside Volatility	Sortino Ratio	Maximum Drawdown
Ridge + MRR + MILP	32.89%	18.07%	4.63%	3.62	2.03%	8.25	−2.88%
LQD Benchmark	19.06%	10.73%	8.37%	1.26	4.89%	2.16	−5.72%

Metrics return from simulating 4 million trades

MONTHLY WALK-FORWARD TRADE-LEVEL AUTOMATION RESULTS													
	model	total_oos_trade_points	automated_trades	automated_dates	automated_bonds	automation_rate	mean_auto_return_bps	median_auto_return_bps	auto_hit_rate	mean_auto_divergence	auto_divergence_volatility	p95_auto_divergence	false_automation_rate
0	Price divergence - AXOR liquidity	4356778	346792	98	16719	0.0796	8.3687	5.8393	0.5661	0.2018	0.0053	0.2091	0.0612
1	Price divergence - TRACE MRR liquidity	4356696	417784	118	16185	0.0959	-0.1535	-0.2249	0.4983	0.1983	0.0090	0.2087	0.0932

Metrics return for Sharp Ratio from testing

auto_divergence_volatility	p95_auto_divergence	false_automation_rate	automation_sharpe_proxy
0.0053	0.2091	0.0612	2.1482
0.0090	0.2087	0.0932	-0.0888

Figure 9 We showed that we can automate 300k trades into automated trades with a high sharp ratio of 2 percent while alternative widely used MRR liquidity model remained -.08 percent.

AXOR liquidity Model implemented in Jaccard Similarity Portfolio Optimization

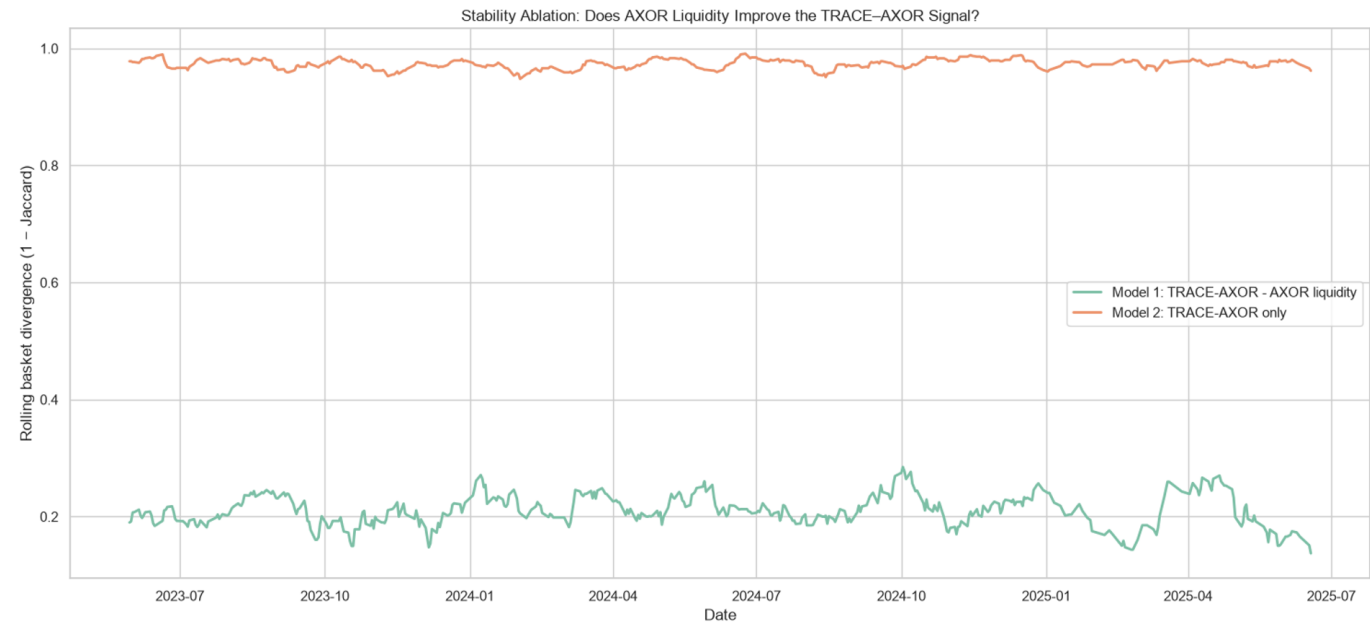
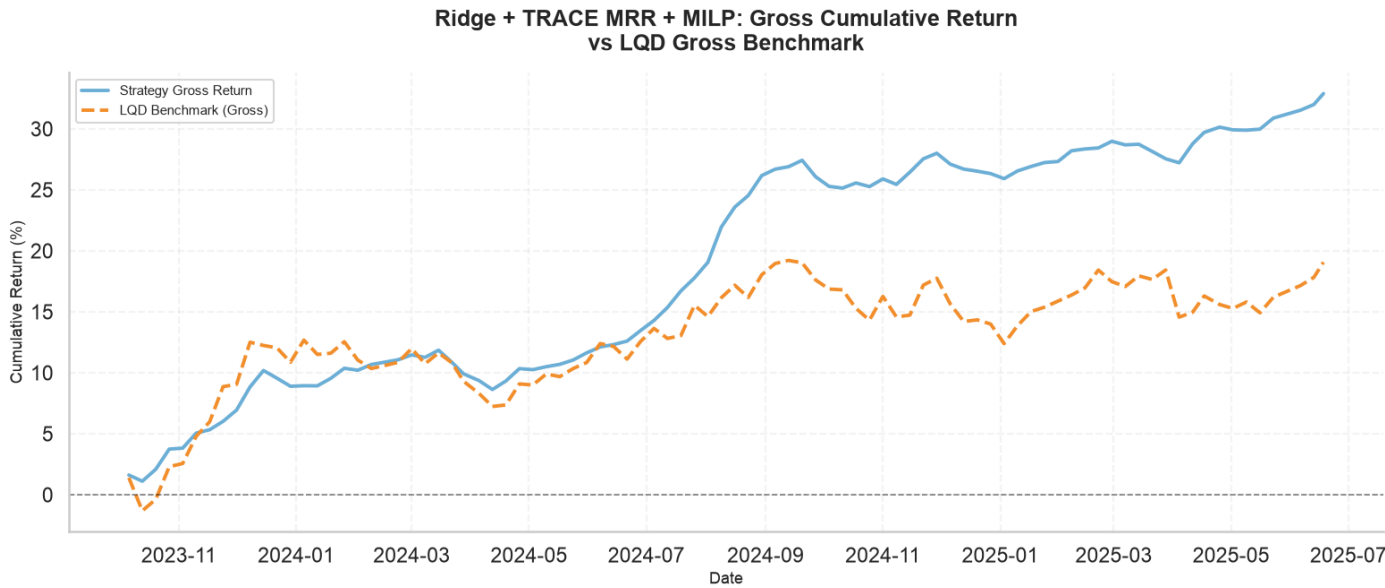


Figure 10 When implementing DeepMM’s internal liquidity model, the Jaccard simailirty for daily portfolio stabilizes with diver-gence staying within range of 0.2 from 0.8

Metrics return for Sharp Ratio from testing specifically for Threshold Design



Metrics Measurement using Ridge Prediction considering holistic trades

STRATEGY VS LQD – RISK ↴ RETURN METRICS							
	Cumulative Return	Annualized Return	Annualized Volatility	Sharpe Ratio	Downside Volatility	Sortino Ratio	Maximum Drawdown
Ridge + MRR + MILP	32.89%	18.07%	4.63%	3.62	2.03%	8.25	-2.88%
LQD Benchmark	19.06%	10.73%	8.37%	1.26	4.89%	2.16	-5.72%

Figure 11 We showed that we can automate 300k trades into automated trades with a high sharp ratio of 8 percent while the bench-mark standard LQD has a negative Sharp during these tested timeline

(a) Threshold Design Testing Timeline and Volatility Metrics

ACTIVE-PERIOD COMPARISON

Active periods: 22

Average exposure while active: 2.71%

V2 active-period volatility: 0.39%

Matched LQD active-period volatility: 0.20%

V2 downside deviation: 0.06%

Matched LQD downside deviation: 0.15%

(b) Model-Specific Cutoff Results

Persistent-Rank Decision Thresholds: AXOR Uncertainty vs TRACE MRR

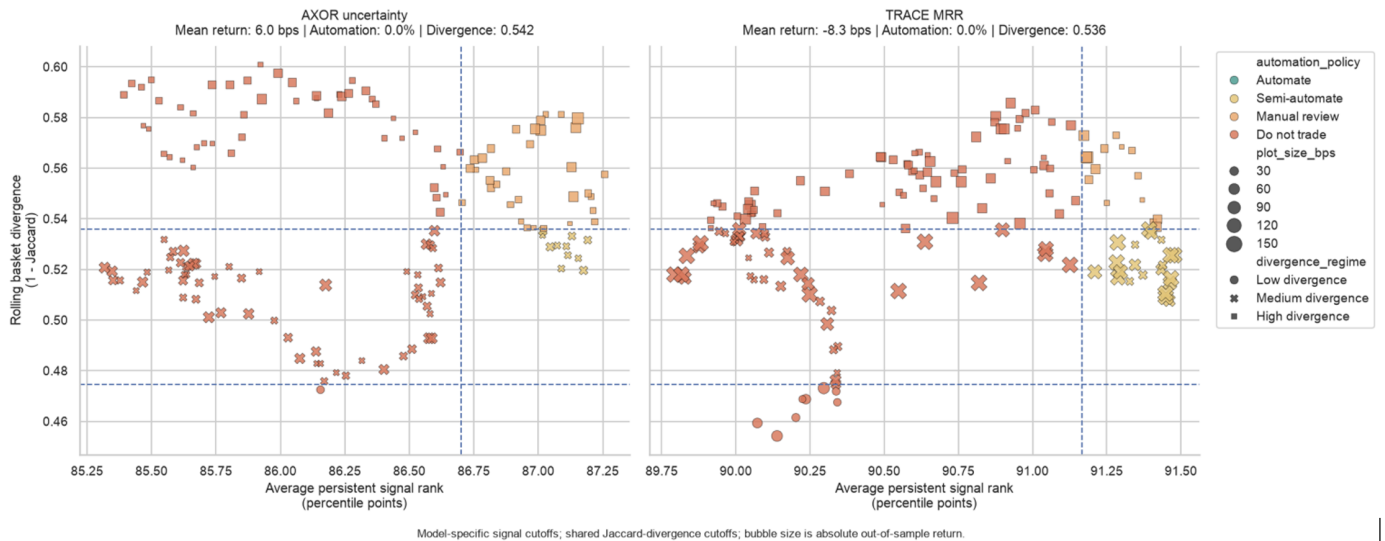


Figure 12 Threshold-design and model-specific performance results. Panel (a) presents the testing timeline and volatility metrics. Panel (b) shows that the AXOR model produces a mean return of 6.0 bps, compared with -8.3 bps for the TRACE MRR model.

(a) Threshold Design Testing and comparing Sharp Ratio with Benchmark LQD

===== RISK-ADJUSTED ACTIVE-PERIOD COMPARISON =====

Advanced V2 Sharpe: 8.173
 Matched LQD Sharpe: -0.419

Advanced V2 Sortino: 56.498
 Matched LQD Sortino: -0.552

(b) Threshold vs Benchamrk Sharp Ratio Comparison

===== ACTIVE-PERIOD COMPARISON =====

Active periods: 22
 Average exposure while active: 2.71%

V2 active-period volatility: 0.39%
 Matched LQD active-period volatility: 0.20%

V2 downside deviation: 0.06%
 Matched LQD downside deviation: 0.15%

Figure 13 Threshold-Design Timeline tested for 22 weeks indicating sharp ratio of 8 while Benchmark LQD shows negative in sharp ratio.