Inverse RL Shrinkage: Learning Expert Demonstrations of Risk-Optimized Portfolios

https://github.com/ashnvael/qamsi

Viacheslav Buchkov, Nikolai Kurlovich
Quantitative Asset Management & Systematic Investments
University of Zürich

June 26, 2025

Table of Contents

Introduction

Motivation (1/2)

- Global Minimum Variance portfolio optimization is a well-studied paradigm that is important for the investors with Min Variance utility, i.e., ones that **seek to minimize the risk, while still allowing for equity premia collection**
- Moreover, in practice Minimum Variance portfolio might prove to be quite a well-performing strategy even in a risk-return space, as **covariances are usually more stable through time**, compared to the mean vectors
- However, the covariance matrix usually requires **some degree of regularization** due to noisy and unstable correlations (and sometimes lack of past data for estimation)

Motivation (2/2)

- De Nard and Kostovic 2025 propose a novel method for controlling such a regularization in a data-driven manner - picking the shrinkage intensity not from the asymptotic optimality, but **learning to efficiently recognize the suitable shrinkage for the current market regime**
- We aim to improve the results of the original paper by introducing a more **robust Reinforcement Learning framework to highly varying environment dynamics in fully out-of-sample construction**

Minimum Variance Portfolio

- The global minimum variance (GMV) portfolio provides a "clean" testbed for evaluating covariance estimators, since it abstracts from expected-return estimation and directly penalizes out-of-sample portfolio volatility.
- In absence of short-selling constraints - we use the FOC-based solution of Quadratic Programming under linear budget constraints:

$$\hat{w} = \hat{\Sigma}^{-1} A^T (A \hat{\Sigma}^{-1} A^T)^{-1} b$$

where $\hat{\Sigma}$ is estimated covariance matrix and budget constraint is given by $Aw = b$

Linear Shrinkage

Estimate covariance by blending sample covariance S and some target F:

$$\tilde{\Sigma} = \delta F + (1 - \delta) S.$$

Optimal δ minimizes

$$\mathbb{E}\|\delta F + (1 - \delta) S - \Sigma\|_F^2,$$

with analytically derived shrinkage intensity (under i.i.d. assumption)

$$\hat{\delta} = \min\left\{1, \max\left\{\frac{\left(\sum_i \sum_j \frac{1}{T}\sum_t ((r_{ij} - \bar{r}_{i\cdot})(r_{ji} - \bar{r}_{j\cdot}) - s_{ij})^2\right) - \hat{\rho}}{T\sum_i \sum_j (f_{ij} - s_{ij})^2}, 0\right\}\right\}$$

Keeps $\tilde{\Sigma} \succ 0$, *which allows for stable inversion.*

This hyperparameter δ is exactly **what we want to learn in the data-driven manner**, i.e. obtaining $\hat{\delta}_\theta(X)$ from market features X (De Nard and Kostovic 2025).

Reinforcement Learning

Setup

- As outlined in the original De Nard and Kostovic 2025 paper, here we work in the 1-step MDP setup with $\mathcal{A}$ the set of actions (the shrinkage intensity) and $\mathcal{R}$ the reward function (the negative of 21-days-ahead realized portfolio standard deviation)
- As usually in finance our actions do not affect the state transition probabilities, we have

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t), \quad \forall \quad a_t \in \mathcal{A}$$

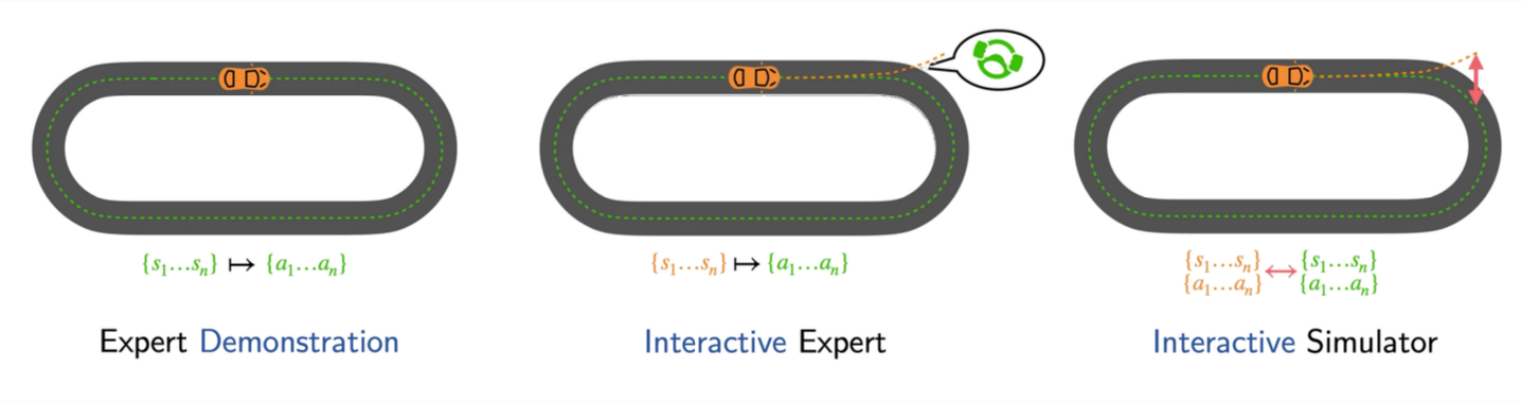
- By the independence of state-visitation distribution d^{π_θ} from the actions a_i we get

$$\min_\theta \mathbb{E}_{a_i \sim \pi_\theta(\cdot|s_i)} \sigma_T^2(\{s_i\}_i^T, \{a_i\}_i^T) = \min_\theta \mathbb{E}_{a_i \sim \pi_\theta(\cdot|s_i)} \sum_i^T \sigma_i^2(s_i, a_i)$$

$$\iff a_i^* = \underset{\theta}{argmin}\ \sigma_i^2(s_i, a_i^\theta) = \underset{\theta}{argmin}\ \sigma_i(s_i, a_i^\theta)$$

Imitation Learning

- However, contrary to the classical RL approaches, in this task the reward design proves to be quite a complex task, as **(negative) 1-month-ahead future realized volatility has non-stationary distribution**
- **Idea**: Learn from optimal shrinkage demonstrations
- **Setup**: For each trading day t we have set of features X_t, optimal demonstration $\hat{\delta}_t^{RL-*}$
- In training our model **is allowed to interact with the environment** to receive the feedback for its action



Behavioral Cloning

Behavioral Cloning

- De Nard and Kostovic 2025 suggest the following approach (Equation 2.12):

$$\pi^*(s) = \underset{\pi \in \diamond}{argmin}\ \mathbb{E}_s[(\pi(s) - y^*(s))^2]$$

- One can instantly recognize the approach as Behavioral Cloning, where we try to learn the **optimal policy by supervised learning**
- The problem (MLE) can be reformulated as KL-Divergence minimization

$$\min_{\theta}\ \mathbb{E}_{s \sim d^{\pi_E}, a \sim \pi_E(\cdot|s)} \left[\log \left(\frac{\pi_E(a|s)}{\pi_\theta(a|s)} \right) \right]$$

- This approach works reasonably well under **static environments** and high number of representative expert demonstrations
- The map $s_t \rightarrow y_t^*$ is modeled with Elastic Net, Random Forests, Gaussian Processes, and Deep Learning (2-layer MLP)

Feature Set

Feature	Description
Average Correlation	Average pairwise sample correlation between stocks.
Average Volatility	Average one-year sample volatility of all individual stocks.
EWMA	Exponentially weighted moving average of previous-month returns (decay $= 0.1$) of the equally-weighted portfolio.
Lagged Optimal Action	Optimal action from the previous rebalancing.
Linear Shrinkage	Linear shrinkage intensity of Ledoit and Wolf (2004b).
Momentum	Fraction of days each stock had positive returns over the previous month, averaged across all stocks.
Rolling Optimal Action	One-year rolling average of the optimal action.
Rolling Optimal SD	One-year rolling standard deviation of the optimal actions.
Trace	Trace of the sample covariance matrix.
Universe Volatility	One-year sample volatility of the equally-weighted universe.

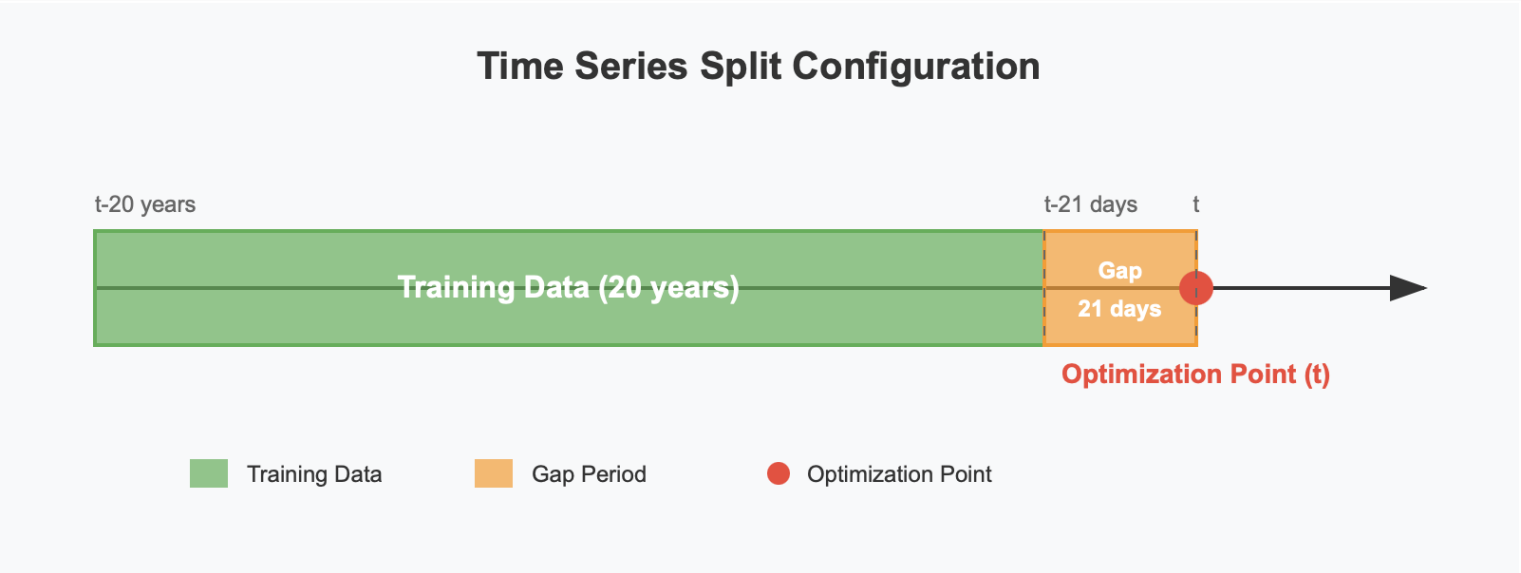
Table 1: Summary of features used.

Black-Box Optimization

- We **follow the methodology** of De Nard and Kostovic 2025 paper and construct our BC target as a shrinkage $\hat{\delta}^{RL-*}$that minimizes 21-days-ahead portfolio standard deviation
- Instead of a search over a pre-defined grid of values, we **search within the domain** of $[\delta_{min}^{RL-*}; \delta_{max}^{RL-*}]$
- This optimization approach is known as "black-box-optimization"
- We employ the **GP-UCB method** (covered further) and check the results with the Golden Section Optimization via grid
- The GP-UCB approach does not require any assumption on the $\sigma_{t:T} : D \to \mathbb{R}^+$ behavior, except for the P-a.s. **L-smoothness**
- We **iterate at maximum** $S = 10$ times for each trading day in the sample

Causal Window

- For each rebalancing date t, targets $\hat{\delta}_t^*$ **depend on realized returns** over $[t,\ t + 21\ \textit{days}]$.
- To avoid look-ahead, we only train on $[t - 20\,\text{years},\ t - 21\,\text{days})$.
- Features at each historic point use only data *up to* that date.
- At prediction time (red), use features as of $t - n$ (trading lag), where lagged target metrics are computed with the causal window too



Green: training window, *Orange*: Causal Window, *Red*: prediction point

Inverse Reinforcement Learning

Pitfalls of Behavioral Cloning

However, Behavioral Cloning has several issues that affect the optimal performance:

1. Long-Term Planning - not accounting for the H-step ahead decision-making policy.
2. Cascading Errors - accumulates the errors over the H horizon, which grows with the horizon length.
3. Distribution Shift - suffers from the case of mismatching training and testing distributions, known as *distribution shift* (Abbeel and Ng 2004, Ziebart et al. 2008)

Pitfalls of Behavioral Cloning

However, Behavioral Cloning has several issues that affect the optimal performance:

1. $\times$ Long-Term Planning - Our actions do not affect state visitation distribution, i.e. we do not need to plan actions, based on which state we will end up in next.
2. $\times$ Cascading Errors - Our decision-making is for $H = 1$ step in the future, as portfolios at each rebalancing period are independent.
3. $\checkmark$ Distribution Shift - Markets represent constantly changing and highly varying environments

It means that while we should aim to approximate our **policy reward** to be close to an expert, Behavioral Cloning achieves **policy behavior** to be close to it.

Solution - Inverse RL

- **Idea**: Given the Expert demonstrations and interactive environment, **learn a Reward Function, for which then optimize the policy**
- **Crucial Assumption**: Expert demonstrations are **the most optimal for each state** (satisfied by construction)
- This approach **avoids the distribution shift problem** (Abbeel and Ng 2004, Ziebart et al. 2008)
- It allows for **optimal decision-making under uncertainty**, i.e. under unknown mapping of shrinkage to future (unknown) portfolio volatility

	IRL	RL
Input	Expert Demonstrations	Reward Function
Output	Optimal policy Reward function	Optimal Policy

Figure 1: : Source: Prof. Dr. Niao He

Our Approach

Model	Description
GAIL	GAN-inspired model that trains distrimunator to distinguish the taken action by the new policy from the Expert Action (Ho and Ermon 2016). Produces more optimal theoretical bounds than BC (Xu, Li, and Yu 2020).
AIRL	An crucial for us improvement of GAIL - it is able to **generalize well under significant variation in the environment** seen in training, recovering **reward function (and thus the nearly-optimal policy) that is robust to changes in dynamics**

- We use 2-layer MLP with 32 hidden neurons as the Reward Net
- The policy is modeled by the PPO (discretized action space)
- We train the model only on the initial dataset before 12/18/2000, and then use it in the inference mode, feeding way less data in the model, compared to BC

Empirical Analysis

Data and Portfolio Construction

- We follow the methodology of De Nard and Kostovic 2025
- Daily CRSP stock returns starting on January 1, 1980 through July 31, 2024 (thus, expanding the original paper's set by two years)
- Filter stocks by being listed at NYSE, AMEX, and NASDAQ (CRSP exchange codes 1, 2, 3)
- Subset of only ordinary common shares (CRSP shares codes 10 and 11)
- We allocate **all data before 12/18/2000 for initial training** (or full training in case of IRL approaches), and use the remaining 5,932 days (283 months) for out-of-sample evaluation
- We use **monthly rebalancing**, choosing month end date as the weights calculation date. Contrary to the original paper, we filter out **only the next day absent stocks**
- We use investable universes of $N \in \{30, 50, 100, 225, 500\}$ portfolio sizes by largest market capitalization at each rebalancing point

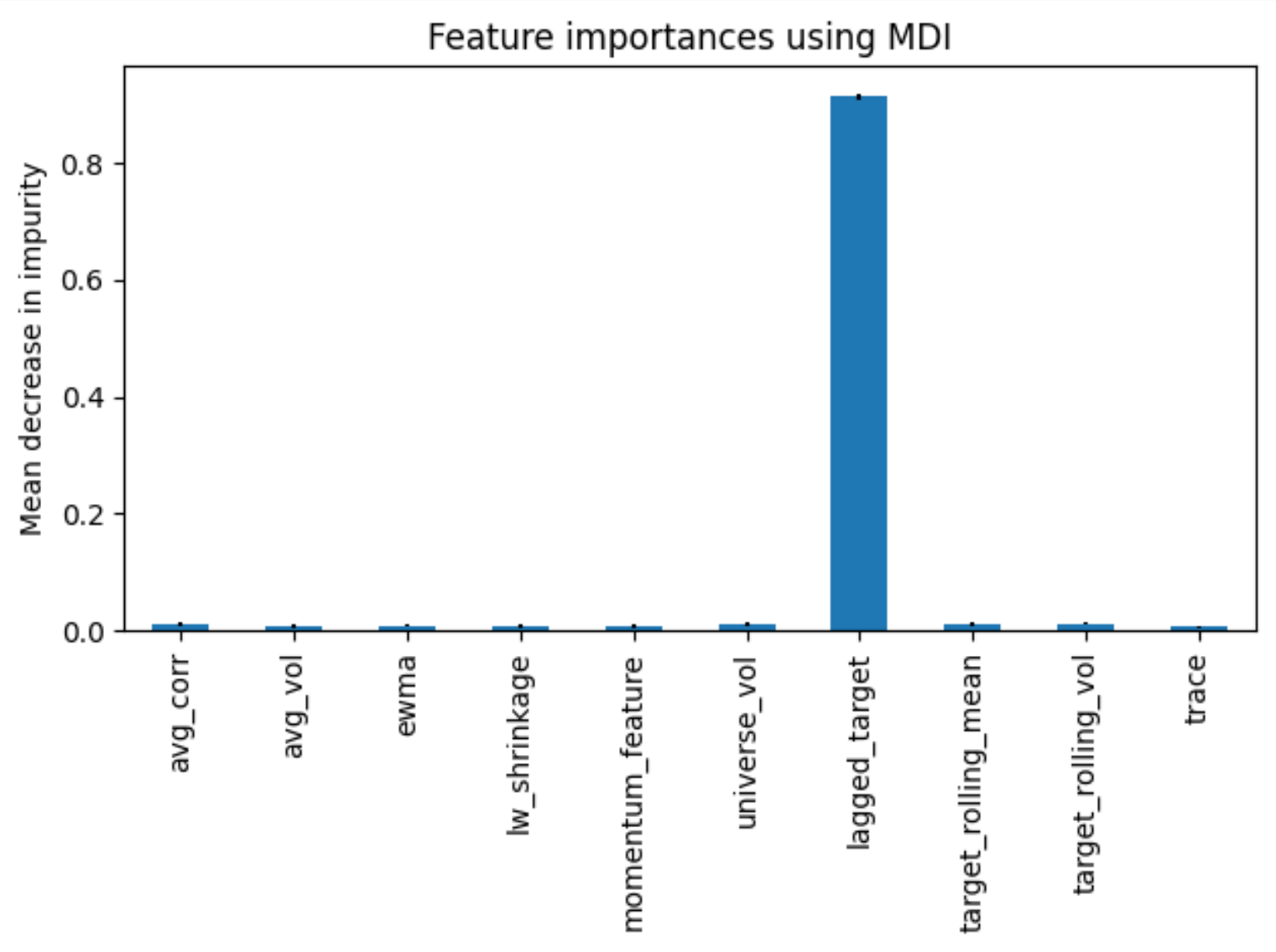
Main Results

Table 2: GMV, (12/2000-07/2024) before TC with monthly rebalancing

	L	F-QIS	DNK	RF	GAIL	AIRL	EW
				SD			
$N = 30$	14.79	14.92	14.47	14.25	14.94	14.09	20.55
$N = 50$	14.58	14.69	14.17	14.06	14.05	14.05	20.68
$N = 100$	14.10	14.13	13.52	13.34	13.16	13.21	21.43
$N = 225$	13.10	13.00	12.44	12.21	12.35	12.43	21.73
$N = 500$	12.30	12.04	11.77	11.61	11.41	11.46	22.66
				SR			
$N = 30$	0.71	0.72	0.77	0.90	0.90	0.94	0.39
$N = 50$	0.73	0.73	0.77	0.86	0.93	0.94	0.40
$N = 100$	0.61	0.60	0.72	0.84	0.82	0.88	0.37
$N = 225$	0.68	0.70	0.78	0.89	0.90	0.93	0.35
$N = 500$	0.69	0.58	0.78	0.82	0.90	0.90	0.37

Feature Importance

Figure 2: Random Forest Feature Importances, $N = 30$



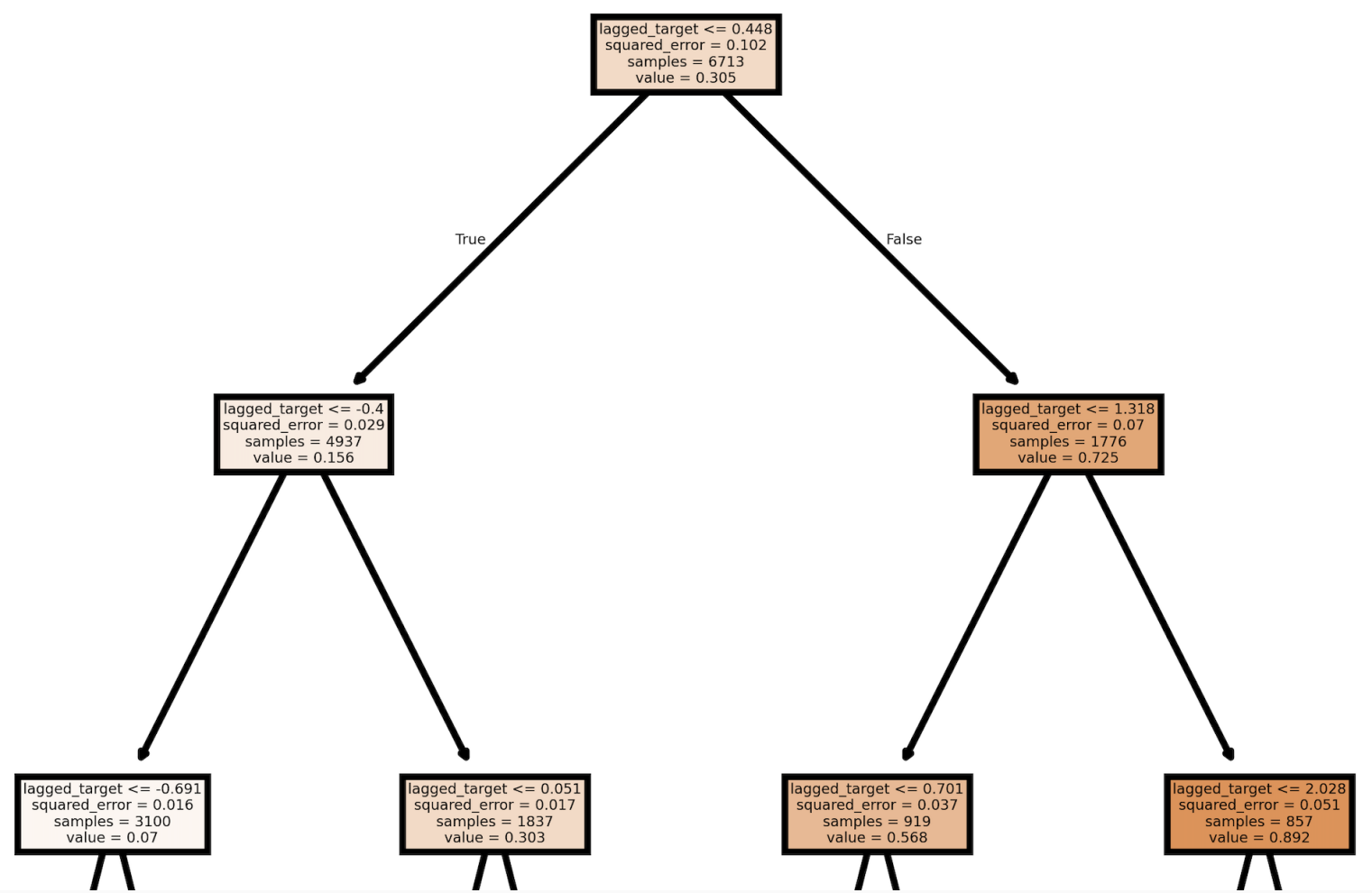
Weak Baselines

Table 3: Weak Baselines Comparison

	MA	Last Optimal	DNK	AIRL
		SD		
$N = 30$	15.80	15.29	14.47	14.09
$N = 50$	15.82	15.48	14.17	14.05
$N = 100$	16.46	15.66	13.52	13.21
$N = 225$	23.27	24.44	12.44	12.43
$N = 500$	15.76	15.45	11.77	11.46
		SR		
$N = 30$	0.71	0.94	0.77	0.94
$N = 50$	0.74	0.74	0.77	0.94
$N = 100$	0.56	0.56	0.72	0.88
$N = 225$	0.31	0.49	0.78	0.94
$N = 500$	0.37	0.41	0.78	0.90

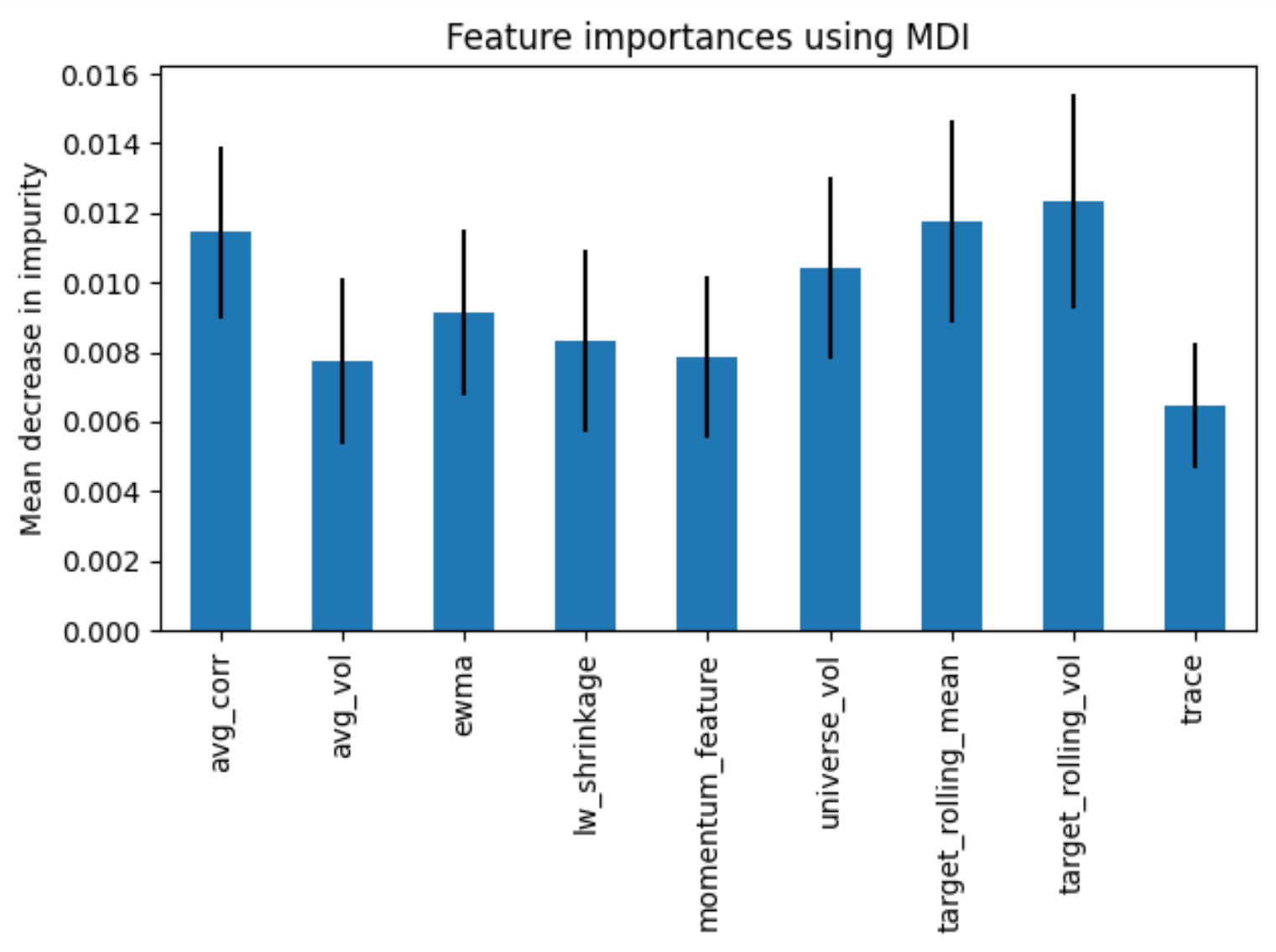
Tree Split

Figure 3: Random Forest (First) Tree Split, $N = 30$



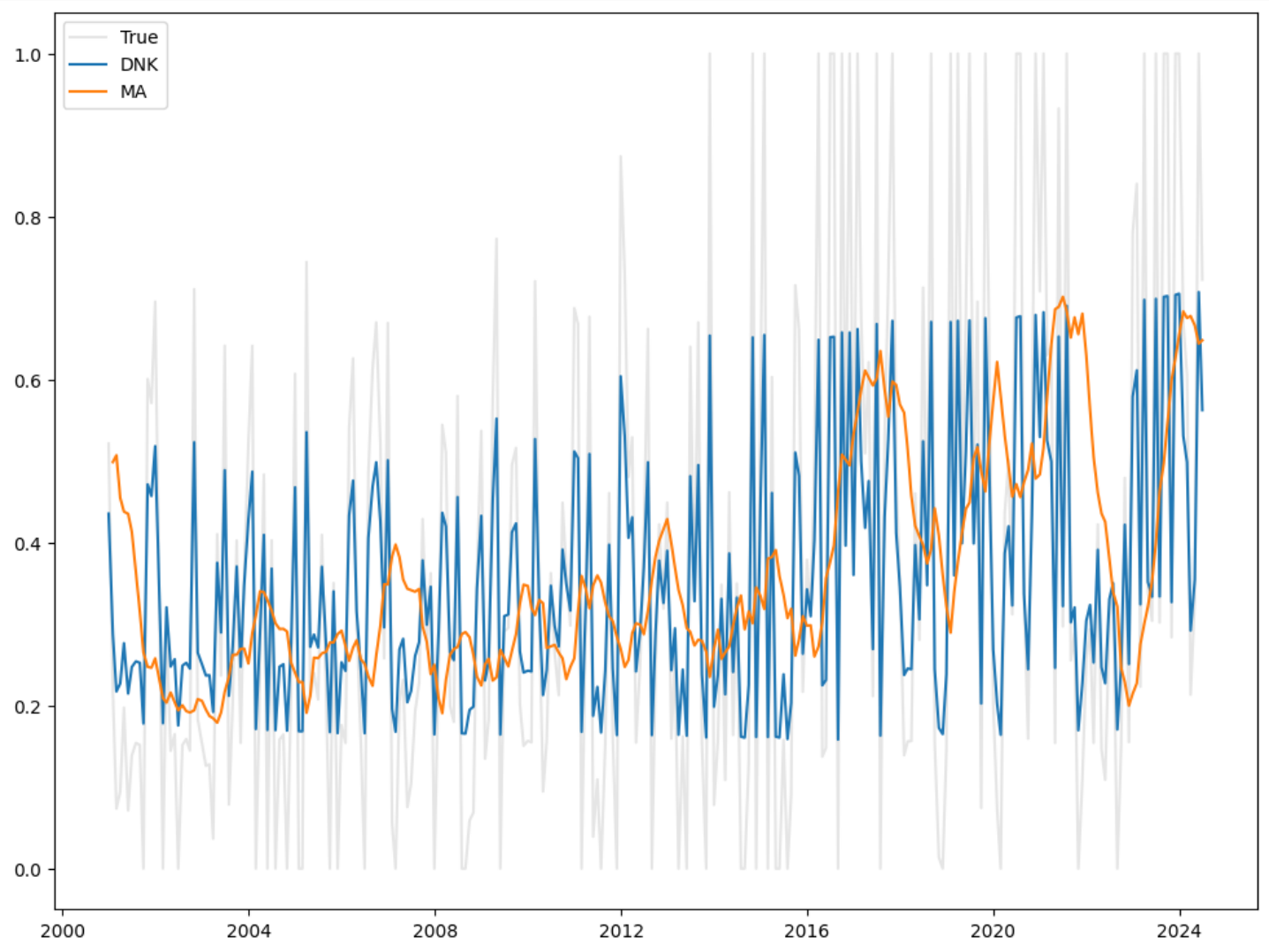
Feature Importance

Figure 4: Random Forest Feature Importances, $N = 30$



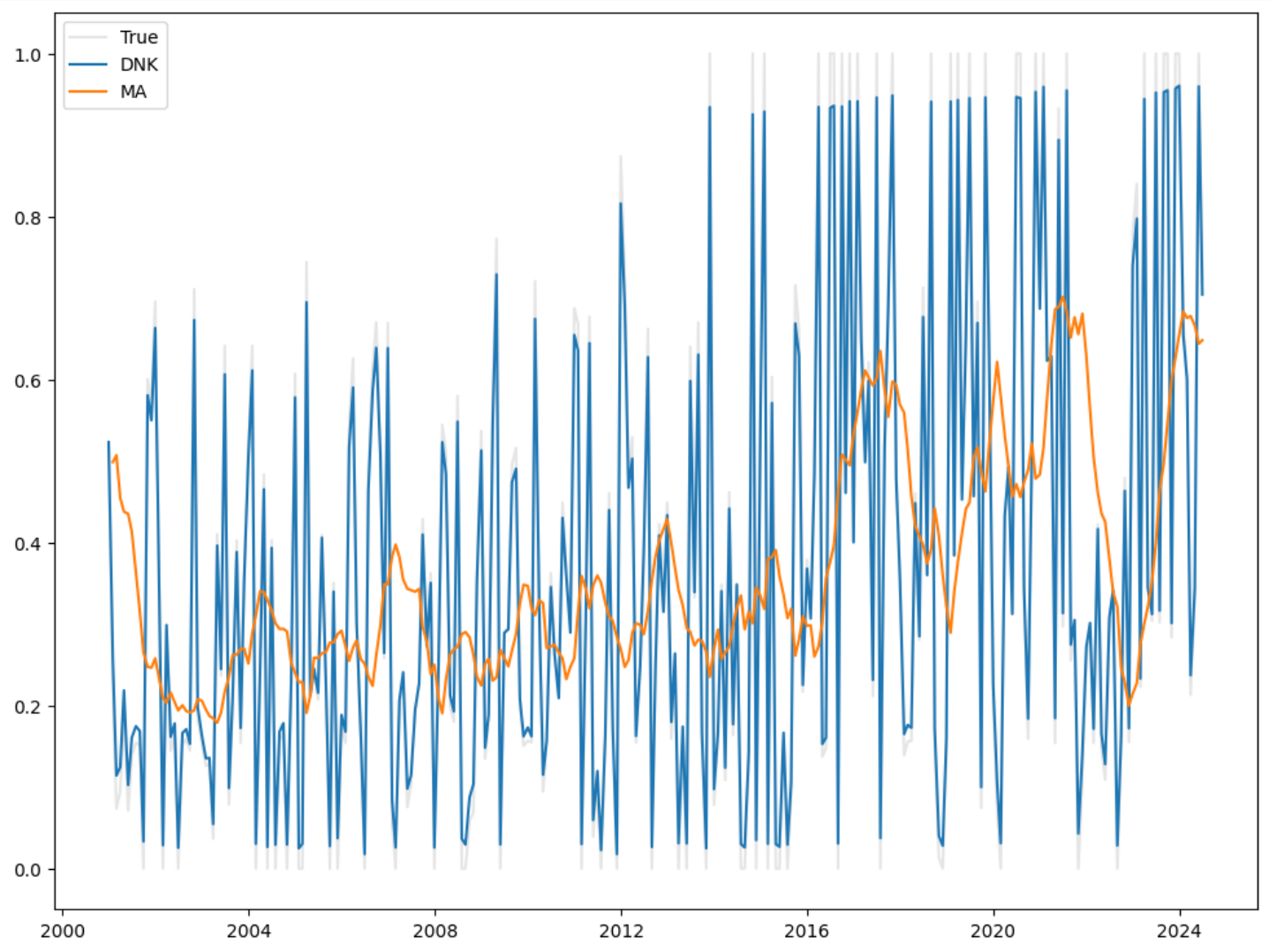
OOS Predictive Performance

Figure 5: De Nard and Kostovic 2025 Approach, $N = 100$



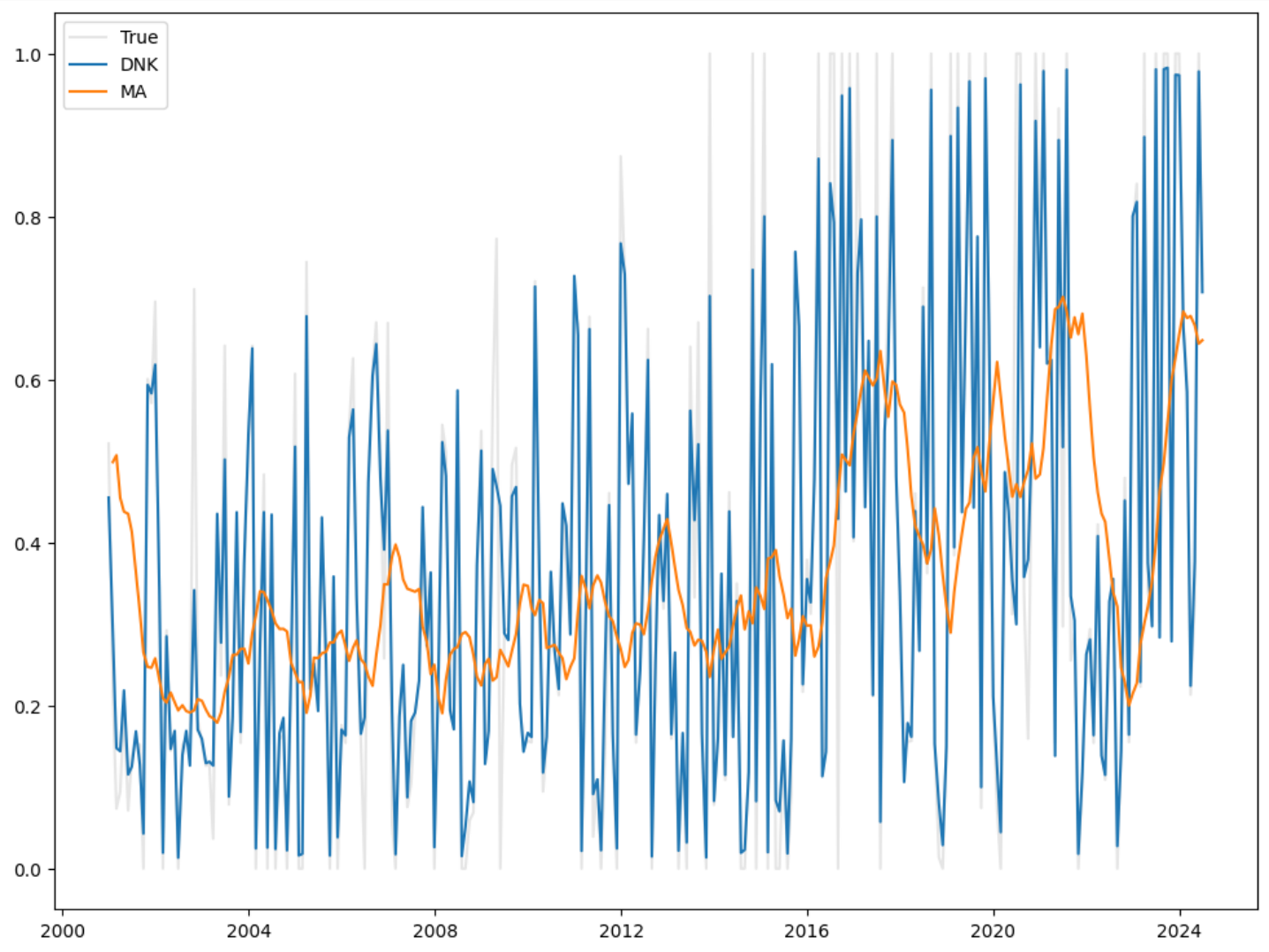
OOS Predictive Performance

Figure 6: OLS Approach, $N = 100$



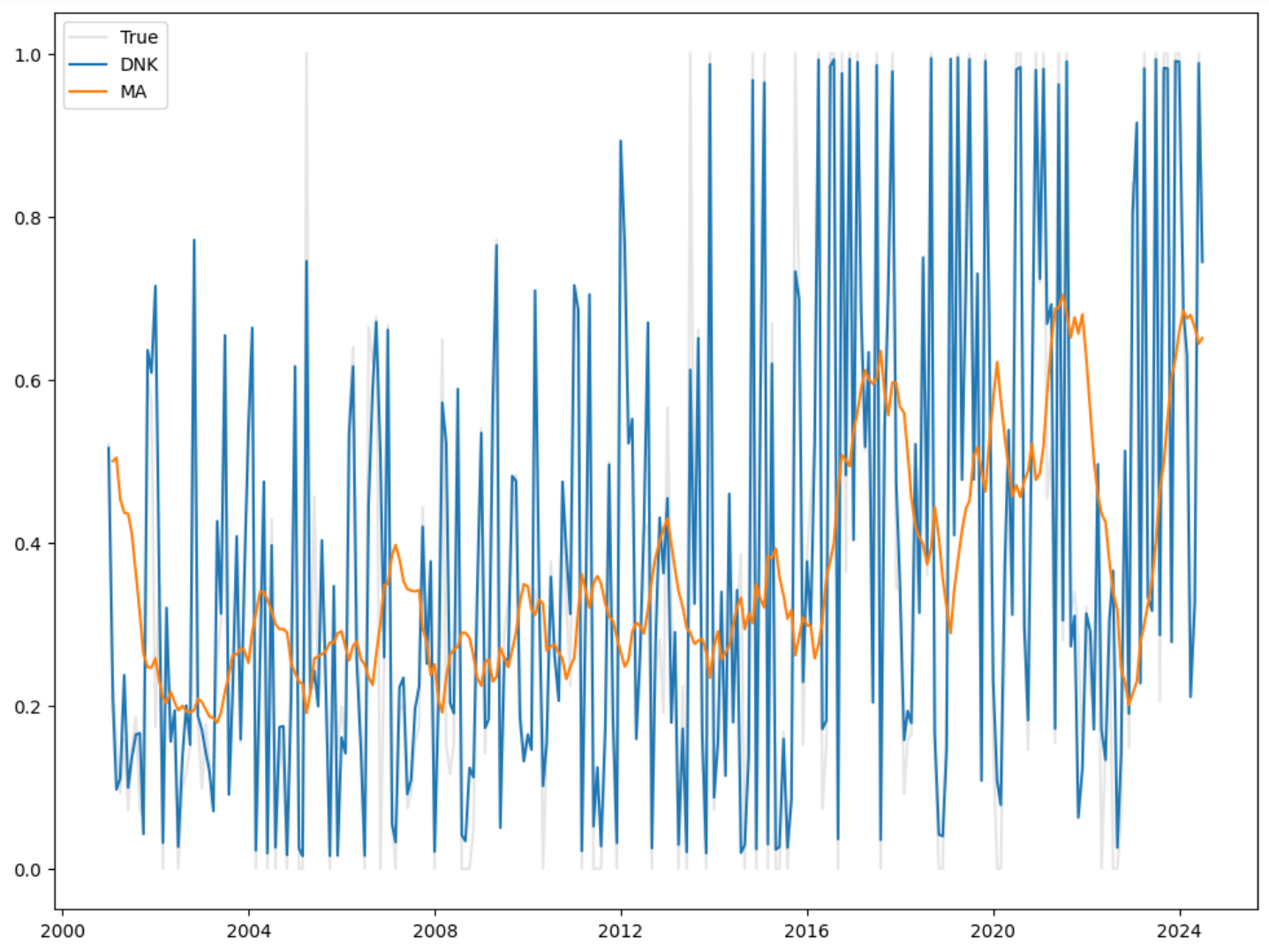
OOS Predictive Performance

Figure 7: Random Forest Approach, $N = 100$



OOS Predictive Performance

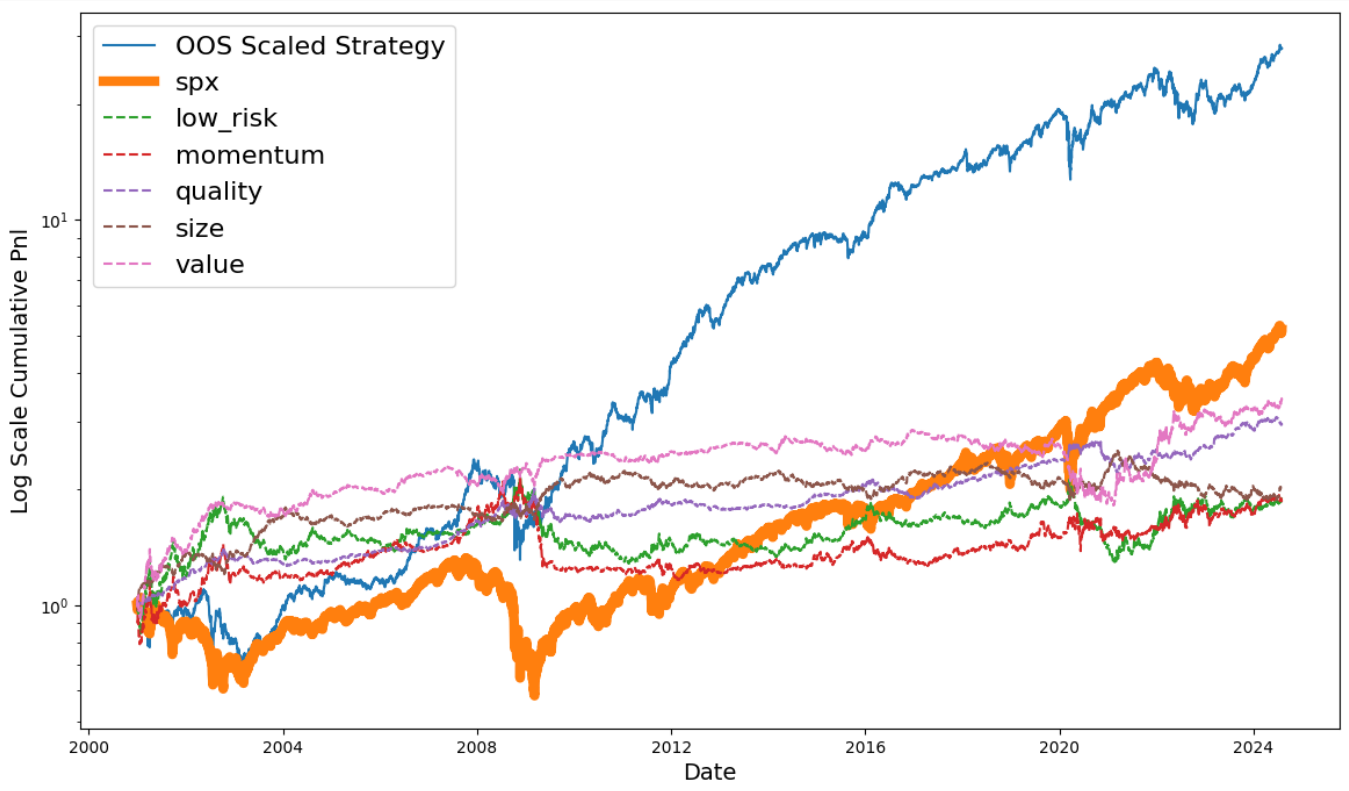
Figure 8: AIRL Approach, $N = 100$



OOS Vol Scaled Strategy Performance

Rescale OOS to match S&P Volatility (Barroso and Santa-Clara 2015)

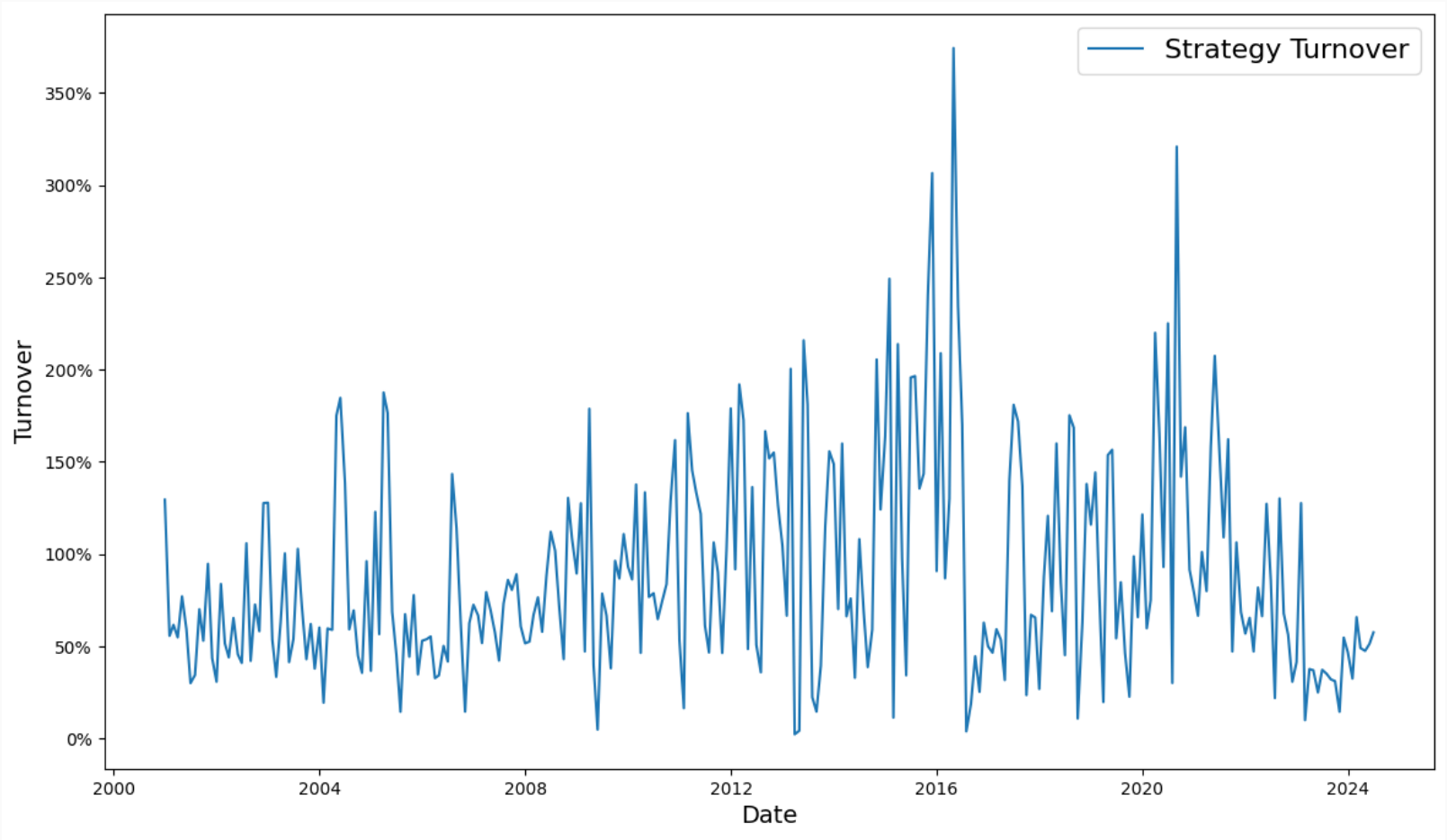
Figure 9: $N = 30$, 5 bps TC, 1 day lag (12/2000-07/2024)



Strategy	SR	Ann. Ret	Vol	MDD	Ann. Alpha	IR
AIRL	0.7674	19.13%	22.41%	-46.05%	7.33% (0.02*)	0.5385
DNK	0.6787	17.17%	22.45%	-47.98%	5.18% (0.07)	0.4108

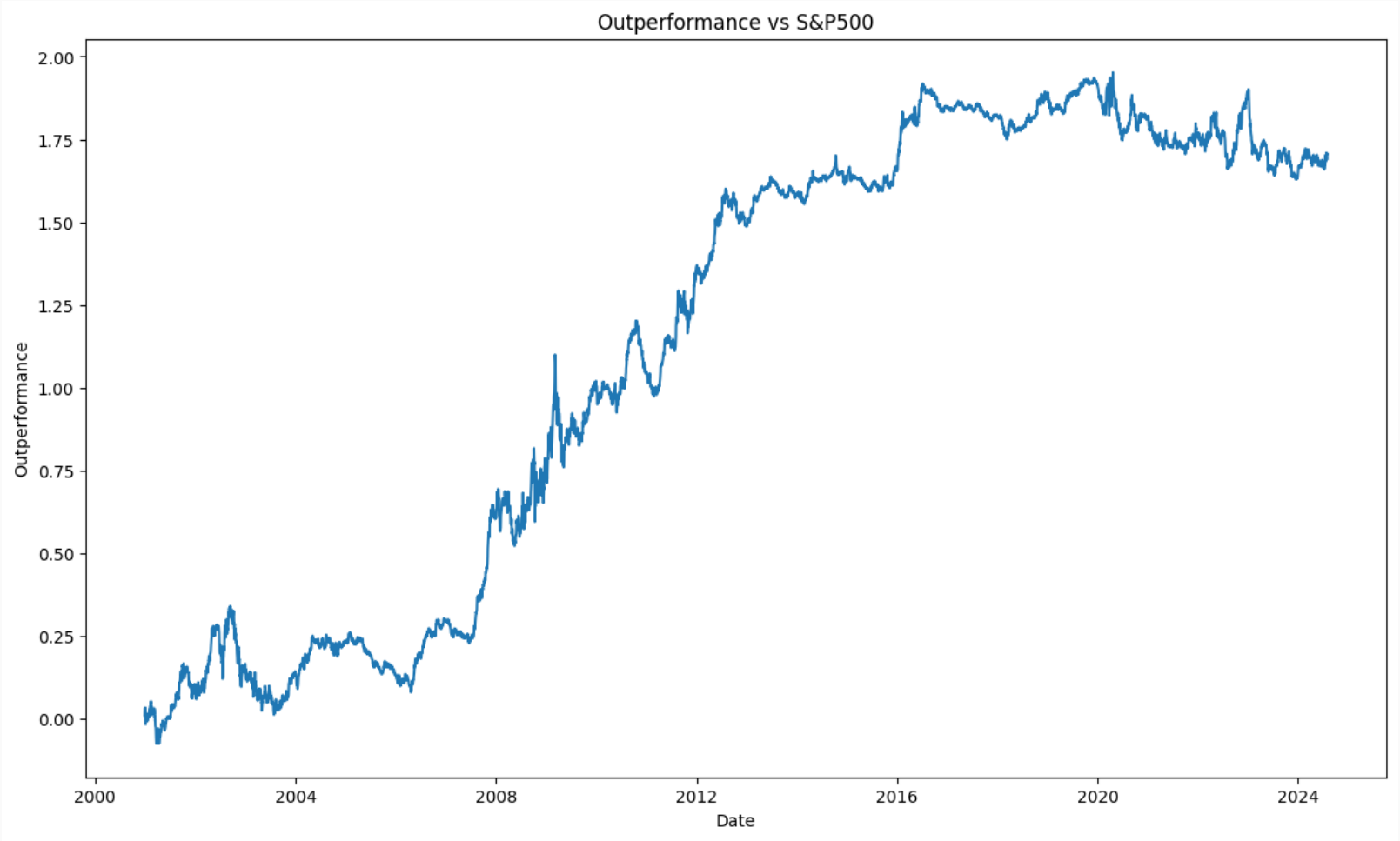
OOS Vol Scaled Strategy Turnover

Figure 10: Turnover (12/2000-07/2024)



OOS Vol Scaled Strategy Outperformance

Figure 11: Outperformance vs S&P500 (12/2000-07/2024)



Conclusion

Future work

- **Testing for QIS shrinkage** - As our paper focuses on Linear Shrinkage only, one should test the IRL approach in RL-NL case too. However, our experiments on $N = 30$ universe shows that the IRL outperformance holds in RL-NL case too.
- **Universal shrinkage learning** - Rather than learning optimal shrinkage separately for each stock universe size N, develop a single model that can learn optimal shrinkage across different N values, using N as an additional feature.
- **Causal window training** - incorporating causal window directly into training to better account for lag effects, which is particularly important for delayed optimal actions.
- **Autoregressive structure** - investigate autoregressive patterns in optimal shrinkage using RNN/LSTM neural networks for both the Reward Net and Policy components, with VAR and Bayesian VAR as comparison baselines.

Conclusion

Project Results

- **Data-Driven Shrinkage** (Initial Paper):
  - We follow the De Nard and Kostovic 2025 methodology (**Behavioral Cloning**) for learning optimal shrinkage intensity
  - Reward is highly non-stationary due to changing market dynamics
    $\Rightarrow$ Avoid reward design by **Expert demonstrations** (original paper)
  - Improve the empirical results by **stronger model** (Random Forest)
- **Inverse RL** (Our Improvement):
  - BC suffers from distributional shifts, IRL is theoretically more suitable approach, which works better due to **robust performance under non-stationary environment**.
    $\Rightarrow$ Learn the reward that "led to having such" Expert demonstration
  - We apply AIRL (Fu, Luo, and Levine 2017) model that is specifically designed for reward learning under **highly varying dynamics**
  - We show empirically that IRL outeperformance comes not from more expressive Risk Minimizer, but **from more optimal learning scheme**

Modeling Framework

All the code for paper replication can be found at the GitHub

```python
from __future__ import annotations

from typing import TYPE_CHECKING

if TYPE_CHECKING:
    import pandas as pd

from qamsi.cov_estimators.rl.base_rl_estimator import BaseRLCovEstimator


class NewCovEstimator(BaseRLCovEstimator):  new *
    def __init__(self, shrinkage_type: str, window_size: int | None = None) -> None:  new *
        super().__init__(shrinkage_type=shrinkage_type, window_size=window_size)

        self.last_pred = None
        self.encountered_nan = False

    def _fit_shrinkage(  new *
        self, features: pd.DataFrame, shrinkage_target: pd.Series
    ) -> None:
        ...

    def _predict_shrinkage(self, features: pd.DataFrame) -> float:  new *
        ...
```

Appendix

Contextual Bandits

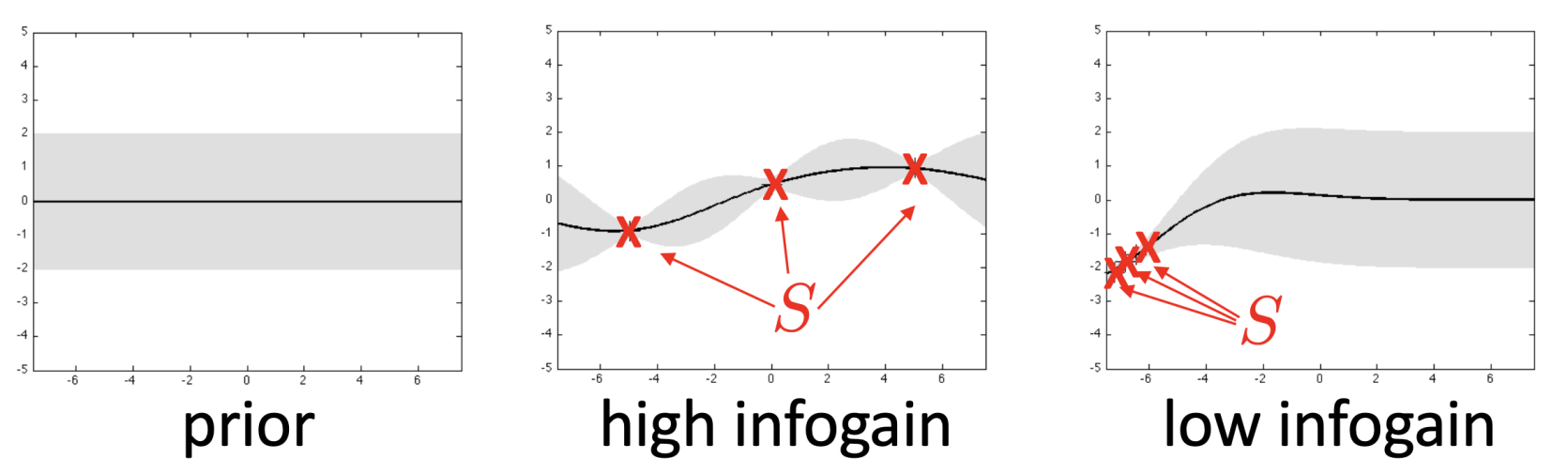
- We test a baseline of learning the optimal shrinkage "in one go", meaning that we want to merge target construction and optimal decision-making under uncertainty
- We scale our reward to have $\tilde{r}_t \in [0; 1]$ by past observed minimum and maximum volatilities for each rebalancing point
- **Given**: domain of shrinkage values $D = \{\delta^{RL-*} \in [\delta^{RL-*}_{min}; \delta^{RL-*}_{max}]\}$, we obtain (potentially noisy) realized standard deviations $\sigma_{t:T} : D \to \mathbb{R}^+$;
- **Task**: Adaptively choose potential maximizers $\delta^{RL-*}_1, ..., \delta^{RL-*}_S$ and observe $\sigma_{t:T}$, where S is the number of iterations.
- This is exactly the contextual bandits setting

CGP-UCB Idea

- We apply the well-known method of Gaussian Process-Upper Confidence Bound (GP-UCB), introduced in Lai and Robbins 1985
- The natural generalization of this method is adding a context (market data features), which still preserves sublinear regret under convex compact decision variable set and (probabilistically) L-smooth function (Krause and Ong Soon 2011)
- We sample a "context" in Time Series manner (i.e. we operate sequentially over each day in the training data).
- The model selects an optimal exploration-exploitation action for this context.
- We evaluate this action, returning a reward, and switch to the next state, meaning the next trading day.
- When we reach a rebalancing date, we "switch-off" the exploration mode (by setting $\beta_t = 0$) and return **pure exploitation** action.

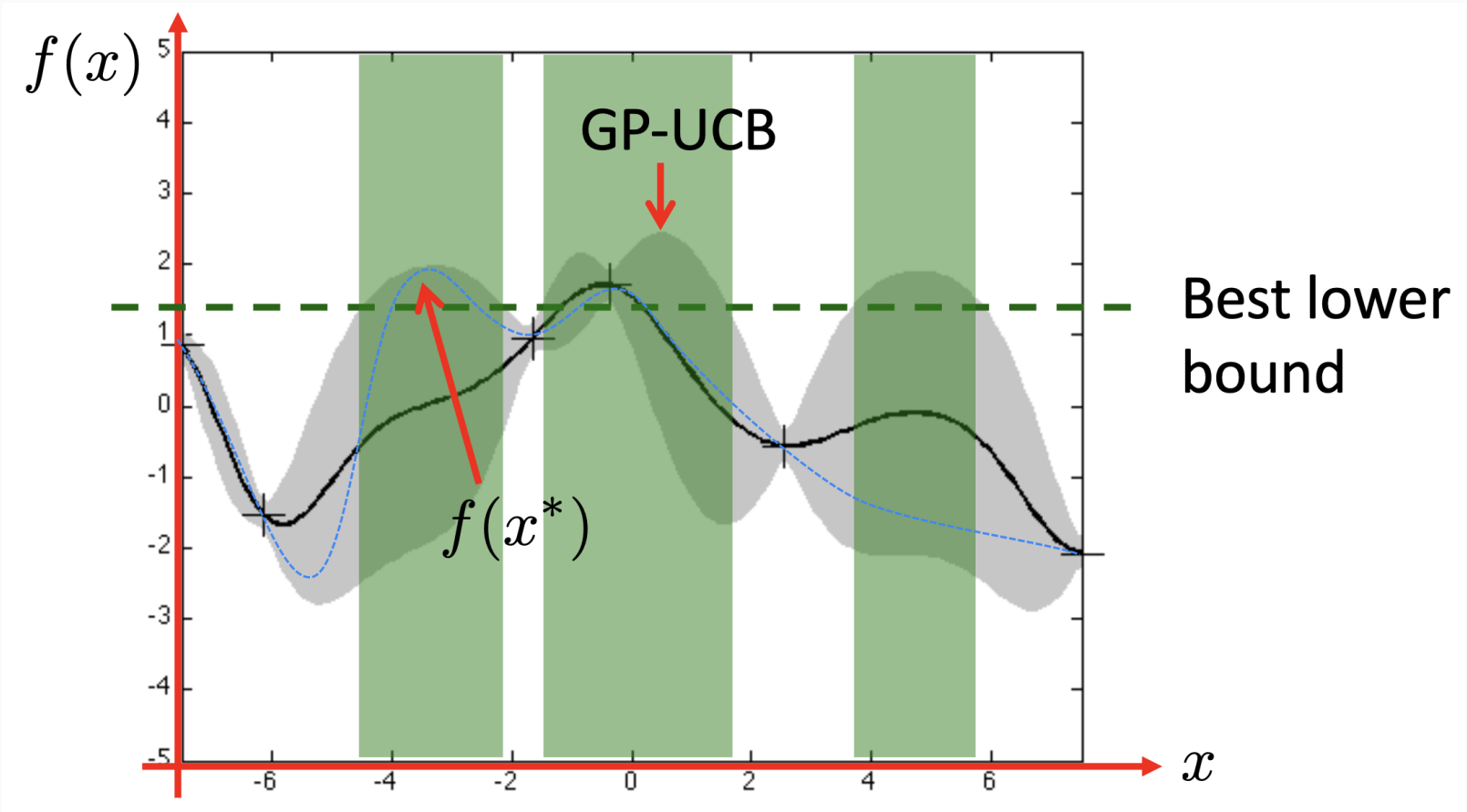
CGP-UCB Demonstration

Figure 12: CGP-UCB Infogain. Source: Prof. Dr. Andreas Krause



CGP-UCB Demonstration

Figure 13: CGP-UCB Algorithm. Source: Prof. Dr. Andreas Krause



Asymptotic Baselines

Model	**Description**
Linear Shrinkage	Ledoit and Wolf 2004 approach.
QIS Shrinkage	Ledoit and Wolf 2020 approach.
Factor Model	Decomposition into low-rank and sparse matrices Residual is handled via: $Diag$, $PCA(k = 3)$ and QIS.

Table 4: Summary of features used.

Elastic Net

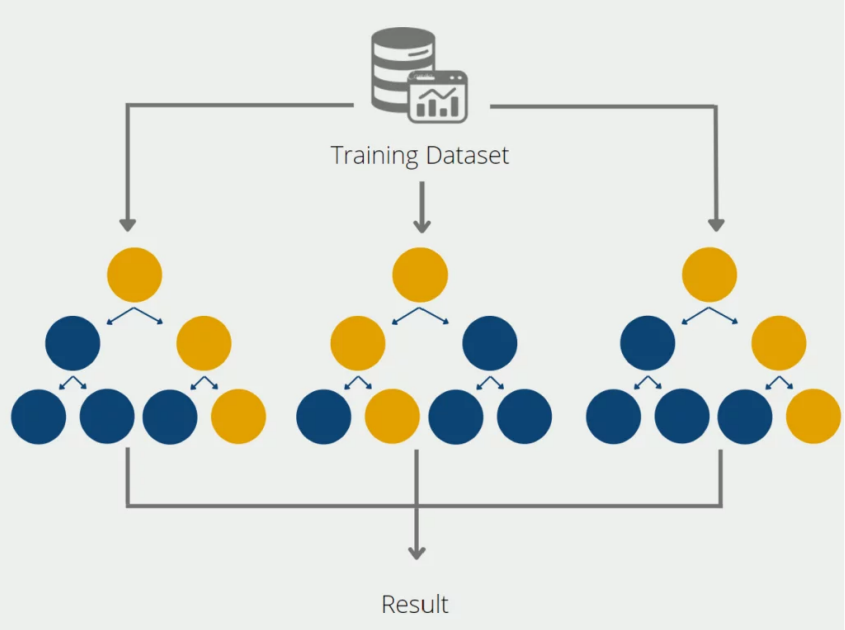
Model: map features s_t to oracle δ_t^* via:

$$\hat{\beta} = \arg\min_{\beta_0, \beta} \frac{1}{T} \sum_t (\delta_t^* - \beta_0 - s_t^\top \beta)^2 + \lambda(\alpha \|\beta\|_1 + \frac{1-\alpha}{2} \|\beta\|_2^2).$$

- Tuned λ, α by time-series CV every 21 days.
- Captures linear relationships with sparsity.
- Unlike De Nard and Kostovic 2025, we tune hyperparameters separately in each rolling window by Time Series Cross-Validation to enforce fully out-of-sample model selection.

Random Forest

- Ensemble of 30 decision trees on bootstrap samples
- At each split, a random subset of features is chosen to minimize MSE
- Captures nonlinear mappings $s_t \mapsto y_t^*$ and guards against overfitting
- No further tuning; robust performance across reasonable settings



Gaussian Process

- Bayesian nonparametric model: places a GP prior (a distribution over functions) over $f : s_t \mapsto y_t^*$ and updates it to a posterior given observed data
- Captures complex nonlinear relationships and provides predictive uncertainty via posterior variance
- Predictive mean:

$$\hat{\delta}(s) \;=\; k(s, S)\left[K(S, S) + \sigma_n^2 I\right]^{-1} y^*$$

- Hyperprior tuning of kernel parameters, with 3 restarts per refit
- Two kernels considered:
  - `DotProduct()` (equivalent to ridge regression)
  - `RBF()` (captures flexible nonlinear structure)

Deep Learning

- Multi-layer perceptron with two hidden layers of size 32, ReLU activations
- Forward pass:

$$h^{(1)} = \mathrm{ReLU}(W^{(1)} s_t + b^{(1)})$$
$$h^{(2)} = \mathrm{ReLU}(W^{(2)} h^{(1)} + b^{(2)})$$
$$\hat{\delta}_t = W^{(3)} h^{(2)} + b^{(3)}$$

- Trained with MSE loss via SGD and cosine-annealing schedule
- Fixed training params:
  $\mathrm{lr} = 10^{-3},\ \mathrm{epochs} = 10,\ \mathrm{batch} = 64,\ \mathrm{weight\ decay} = 0$
- Matches the reward/policy net used in Inverse RL for a direct complexity comparison