SiraEdge - Portfolio Optimization Module

Research and Implementation of Quantitative Models

Ismail Moudden

SiraEdge — Portfolio Optimization Module

August 28, 2025

Abstract

This report presents the portfolio optimization module developed for the SiraEdge platform, an innovative solution aimed at democratizing access to professional asset management tools. We detail the implementation, comparative analysis, and validation of seven optimization models: Markowitz, Risk Parity, Monte Carlo, Black-Litterman, Machine Learning, Hybrid, and Custom Metrics. The study includes a robust walk-forward analysis and performance evaluation on real data (2020-2023). All source codes, figures, and results are available in the associated GitHub repository.

Scope. This document is a **technical and educational report** showcasing the results of research and implementations; it constitutes *neither an allocation recommendation*, *nor a promise of future performance*.

SiraEdge Context and Vision

Our Mission

SiraEdge was born from a simple yet ambitious vision: **to democratize access to professional portfolio management tools**. Our mission is to make finance accessible to everyone by combining mathematical rigor with intuitive user experience.

The SiraEdge Platform

SiraEdge is an interactive platform developed for learning to invest and simulating portfolios. It integrates several modules:

- **Portfolio Simulation**: Testing strategies without risk
- **Optimization Module**: This report presents this module
- **Interactive Learning**: Tutorials and educational resources
- **Performance Analysis**: Advanced metrics and visualizations

Context of this Optimization Module

This document presents **the concrete result of R&D work and implementation** of the optimization models integrated into SiraEdge. The objective is **to showcase the fruits of research** (codes, figures, protocols) and provide a reproducible foundation. This is not an investment recommendation, but a *technical and educational report*.

- Making available ready-to-use scripts for generating figures
- Comparing models on a consistent test universe (2020–2023)
- Explaining risk/return trade-offs and practical limitations
- Documenting hyperparameter choices and backtesting protocols

Positioning in the SiraEdge AI Assistant

In SiraEdge, the AI assistant is **a chatbot for help and analysis**: it answers user questions, explains figures, and **analyzes strategies** using available metrics (Sharpe, Sortino, Calmar, stability, turnover). It can compare models and interpret results, but **it does not provide personalized investment recommendations**. The graphs and tables in this report constitute its technical knowledge base.

Report scope — what it is / what it is not

What it is

- A report on R&D and integration of models (Markowitz, Risk Parity, BL, ML, Hybrid, etc.)
- Support for the AI assistant to illustrate concepts and results
- A reproducible backtesting protocol with standardized metrics

What it is not

- A turnkey trading strategy or personalized recommendation
- A marketing study; the numbers serve *learning* and comparison
- A commitment to future performance; the universe and period are for demonstration

Backtesting and statistics in SiraEdge

The optimization engine is connected to SiraEdge's **backtesting module**. The scripts produce return series and metric tables (Sharpe, Sortino, Calmar, weight stability, turnover) that feed the `Statistics` and `Simulation` pages. The results are presented **for comparative and educational purposes** and may differ from future performance; the numbers depend on the period (2020–2023), the chosen universe, and assumptions (costs, constraints).

Educational Approach

Our approach combines:

- **Scientific Rigor**: Faithful implementation of academic models
- **Accessibility**: Clear explanations and concrete examples
- **Reproducibility**: Complete source code and detailed documentation

GitHub Repository

All source codes, figure generation scripts, and resources for this project are available in the GitHub repository:

`https://github.com/SiraEdge/portfolio-optimization`

Repository Structure:

- `src/`: Python implementation of all models
- `figures/`: Generated graphs and visualizations
- `rapport/`: LaTeX sources for this report
- `README.md`: Installation and usage instructions
- `requirements.txt`: Python dependencies

Important note: This report references specific code files. For complete understanding and the ability to reproduce results, it is essential to consult the associated GitHub repository.

Contents

1 Introduction

This report presents, in an educational and reproducible framework, several portfolio optimization approaches: classical models, machine learning-driven methods, hybrid variants, and optimization through custom metrics. The figures are generated by Python scripts (in the `src/` folder) and referenced here.

Research context and objectives

This document presents a portfolio allocation framework compliant with academic standards: formal model exposition, explicit assumptions, experimental protocol, hyperparameters, results, and critical analysis. The ambition is to produce a *reproducible* artifact (scripts, figures, dependencies) and *interpretable* (justification of choices and limitations).

2 Concepts and Terminology

2.1 Fundamental finance concepts

Portfolio: This is simply a collection of assets (stocks, bonds, etc.) in which you place your money. Think of it as a basket with different fruits - each fruit represents an asset, and the quantity of each fruit represents the weight of that asset in the portfolio.

Return: This is the gain or loss as a percentage over a period. If you buy a stock at $100 and it's worth $110 a year later, the return is 10%. In finance, we often use *logarithmic returns* because they add up better over time.

Volatility: This is a measure of the instability of an asset's price. The more volatile an asset, the more its price can rise or fall rapidly. It's like a car driving fast - it can go far but also skid more easily.

Correlation: Measures how much two assets move together. If two stocks rise and fall at the same time, they are positively correlated. If one rises when the other falls, they are negatively correlated. A correlation of 0 means they move independently.

2.2 Analysis and testing tools

Backtest: This is like "replaying history" with an investment strategy. We take past data and simulate what would have happened if we had applied our strategy at that time. It's useful for testing ideas without risking real money, but beware: the past doesn't guarantee the future!

Walk-forward: A more sophisticated technique than simple backtesting. Instead of testing on the entire history, we test on sliding windows (for example, 252 days = 1 trading year). This better simulates reality where we only know the past to make decisions.

Rebalancing: The action of modifying asset weights in the portfolio to return to the target allocation. For example, if the goal is 50% stocks + 50% bonds, and after a year stocks are worth 60% of the portfolio, we sell stocks to buy bonds.

2.3 Investment styles

Defensive: A strategy that prioritizes stability and capital protection over maximum performance. We accept lower returns to avoid large losses. It's like driving carefully on a slippery road.

Low turnover: A strategy that limits frequent changes in the portfolio. The less we move, the less we pay in transaction costs. It's like a gardener who lets his plants grow rather than constantly moving them.

Core-satellite: An approach that combines two strategies: a stable "core" that represents the majority of the portfolio, and a more dynamic part (satellite) to try to improve performance. It's like having a solid house (core) and an ornamental garden (satellite).

2.4 Performance metrics

Sharpe Ratio: Measures the efficiency of an investment by comparing excess return (relative to the risk-free rate) to volatility. The higher it is, the better. It's like measuring how many kilometers you get per liter of gas - you want maximum return for minimum risk.

Sortino Ratio: Similar to Sharpe, but only penalizes "negative" volatility (losses). This is more logical because we worry more about losses than gains.

Calmar Ratio: Compares annualized return to maximum drawdown (largest loss over a period). The higher it is, the faster the portfolio recovers from its losses.

Maximum Drawdown (MDD): Largest loss as a percentage from a peak. If your portfolio was worth \$100, then \$80, then \$90, the MDD is 20% (from \$100 to \$80).

Weight stability: Measures how much the portfolio allocation remains stable over time. Weights that change little mean fewer transaction costs and a more predictable strategy.

Turnover: Measures trading activity. A turnover of 100% means we traded the equivalent of 100% of the portfolio value over the period. The higher it is, the more transaction costs impact performance.

2.5 Costs and constraints

Transaction costs: Costs paid on each buy or sell (broker commissions, spreads, etc.). Generally expressed in "basis points" (bps): 1 bps = 0.01%. Costs of 5 bps mean we pay 0.05% on each transaction.

Weight constraints: Limits imposed on each asset's share. For example, "no more than 20% in a single stock" or "at least 10% in cash". These constraints often reflect risk management rules or regulatory preferences.

3 Data and Preprocessing

3.1 Assets and period

We use daily prices (2020–2023) for the following assets: ETFs `SPY`, `QQQ`, `IWM`, `EFA`; commodities `GLD`, `SLV`, `USO`, `DBA`. Data is downloaded via `yfinance`.

3.2 Returns and indicators

The **logarithmic returns** of a price P_t are defined by $r_t = \ln(P_t/P_{t-1})$. They add up over time and approximate simple returns well for small variations. **Momentum** here refers to a cumulative short-term variation (e.g., 20 days), **rolling volatility** a local dispersion measure, and **RSI** (Relative Strength Index) a bounded oscillator $[0, 100]$ capturing the speed of rises/falls. Figure 1 shows the correlation between assets.

3.3 Scripts (excerpt)

```
prices = yf.download(tickers=TICKERS, start=start, end=end, auto_adjust=
    True)["Close"].dropna()
log_returns = np.log(prices / prices.shift(1)).dropna()
features = {
  t: pd.DataFrame({
    "momentum": prices[t].pct_change(20),
```

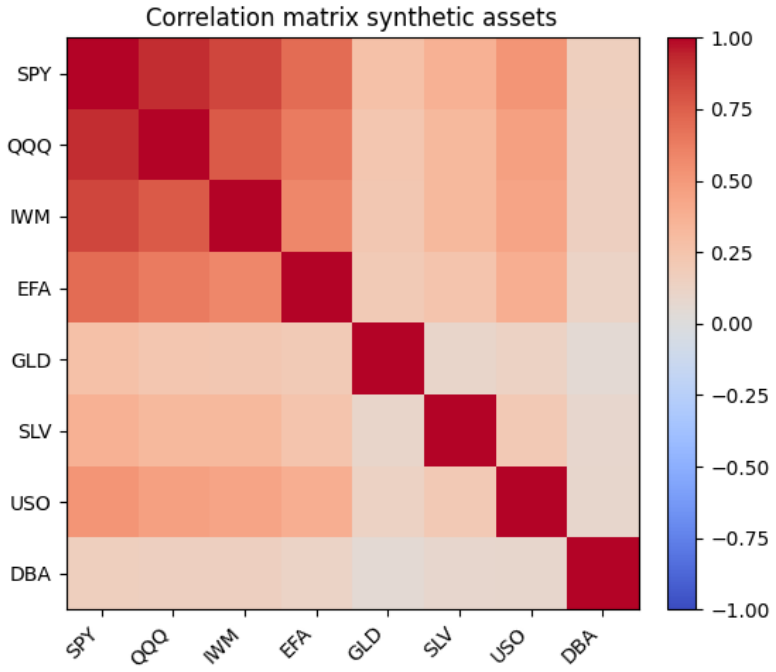


Figure 1: Correlation map of daily returns (2020–2023). Each cell measures the co-variation between two assets (1 on the diagonal). Values close to extremes indicate strong correlation; values close to 0 indicate good diversification potential.

```
    "volatility": prices[t].pct_change().rolling(20).std()*np.sqrt(252),
    "rsi": RSIIndicator(prices[t], window=14).rsi(),
  }).dropna()
  for t in prices.columns
}
```

Listing 1: Download and preprocessing (excerpt)

3.4 Experimental protocol

Unless otherwise stated, statistics (expectation, covariance) are estimated over the entire 2020–2023 window and portfolios are *long only*, fully invested. Reported metrics: annualized return, annualized volatility, Sharpe ratio, and maximum drawdown. A *walk-forward* protocol (252-day sliding window with monthly rebalancing) is also described for future work.

4 Markowitz (Mean-Variance)

4.1 Fundamental principle

Markowitz theory (1952) is the foundation of modern portfolio optimization. The basic idea is simple: **the more risk you take, the more return you can expect, but you need to find the right balance**.

In simple terms: Imagine you want to invest your money. You can put it in a savings account (low risk, low return) or in stocks (higher risk, potentially higher return). Markowitz tells you: "For a given risk level, here's the best combination of assets that maximizes your return."

4.2 Mathematical objective

Optimize weights w that maximize the **Sharpe ratio** (excess return per unit of risk) or minimize variance for a target return. Classical formulation:

$$\min_{w} \; w^\top \Sigma\, w \quad \text{s.t. } \mathbf{1}^\top w = 1,\; w \geq 0.$$

Simple translation: We're looking for the percentages to put in each asset so that:

- The sum of percentages = 100% (constraint $\mathbf{1}^\top w = 1$)
- No negative percentage (constraint $w \geq 0$)
- The "total risk" is as low as possible for a given return

From a formal perspective, in long-only and unit budget, Markowitz optimization is posed as a convex quadratic program: $\min_w \; \frac{1}{2} w^\top \Sigma w - \lambda\, \mu^\top w$ subject to $\mathbf{1}^\top w = 1$ and $w \geq 0$. The solution satisfies KKT conditions and is solved efficiently by QP. In practice, we stabilize Σ by Ledoit-Wolf shrinkage ($\Sigma_\eta = (1-\eta)\,\hat{\Sigma} + \eta\,\mathrm{diag}(\hat{\Sigma})$) and/or a ridge $\Sigma + \varepsilon I$. The "max Sharpe" version under long-only constraint reduces to a QP by fixing λ through linear search or setting a volatility target. The extension with risk-free asset is expressed in second-order cone when volatility is bounded.

What is Σ? It's the covariance matrix that measures how assets vary together. If two assets rise and fall at the same time, their covariance is positive. If one rises when the other falls, it's negative.

Numeric example — Sharpe ratio

Suppose a portfolio with annualized return $R = 8\%$ and annualized volatility $\sigma = 12\%$ (negligible risk-free rate). The Sharpe ratio is $S = R/\sigma = 0.08/0.12 \approx 0.67$. Another portfolio with $R = 10\%$ but $\sigma = 20\%$ has $S = 0.50$: despite higher return, it is *less efficient*. **Key takeaway:** Sharpe compares return to risk; a return increase is not profitable if the risk increase is more than proportional.

4.3 Main assumptions

Gaussian returns, stationary covariance, absence of transaction costs and additional constraints.

4.4 Implementation (excerpt)

```python
# Markowitz optimization with Monte Carlo
def optimize_markowitz(returns, target_return=None):
    """Optimizes a portfolio according to Markowitz theory"""
    # ... existing code ...
```

Listing 2: Markowitz - File: `src/markowitz.py`

4.5 Results

Numeric example — Risk parity with 2 assets

Two assets with $\sigma_1 = 20\%$, $\sigma_2 = 10\%$ and correlation $\rho = 0.2$. In *equal risk contribution*, weights approach the inverse of volatilities when ρ is moderate: $w_1 \approx 1/3$, $w_2 \approx 2/3$. Thus the less volatile asset carries more weight to balance risk contributions. **Consequence:** portfolio often more diversified and more stable than Markowitz when μ is uncertain.

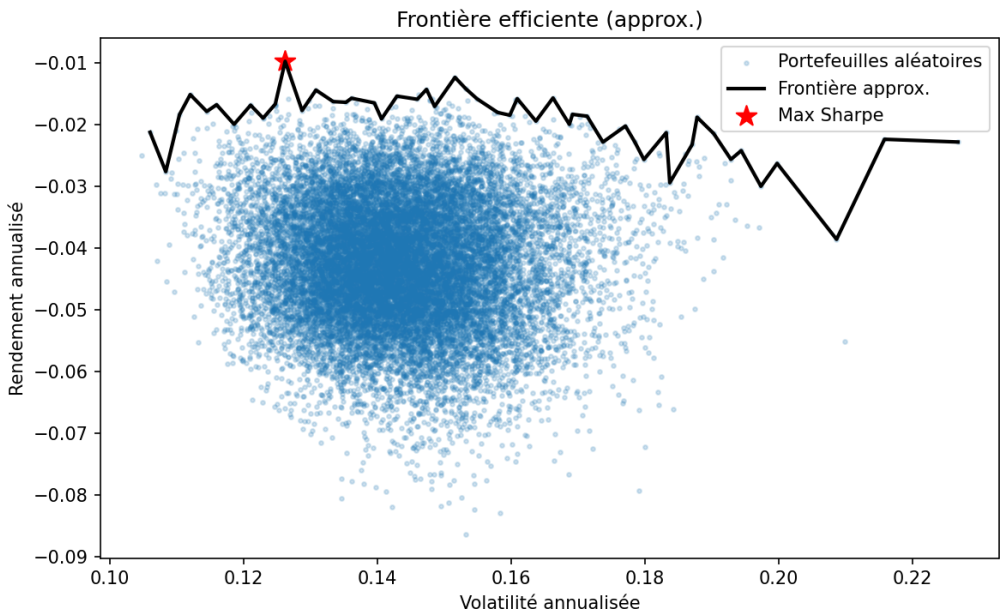


Figure 2: Approximate efficient frontier (black line) in the annualized volatility (x-axis) – annualized return (y-axis) plane. Blue points are random portfolios and the red star indicates the portfolio with the highest Sharpe ratio; higher and to the left is preferred.

4.6 Analysis and limitations

The frontier illustrates the return/risk trade-off. However:

- **Estimation sensitivity**: errors on μ often dominate allocation; shrinking μ or using objectives without μ (min-var, risk-parity) can stabilize.
- **Non-stationarity**: Σ varies over time; prefer sliding windows or robust estimators.
- **Concentration**: in long-only, the portfolio can concentrate on few assets with high estimated expectation.

Practical usage

- Relevant for *liquid universes* and medium/long-term horizons.
- Prefer a *max Sharpe* version with weight constraints and concentration bounds.
- Monthly/quarterly re-estimation; add turnover ceiling if costs.

5 Risk Parity

5.1 Fundamental principle

Risk Parity is an approach that **equalizes the risk contribution of each asset** in the portfolio. Unlike Markowitz which is based on expected returns (very difficult to estimate), Risk Parity focuses solely on risk management.

In simple terms: Instead of saying "I want 50% stocks and 50% bonds", Risk Parity says "I want stocks and bonds to each contribute 50% of the total portfolio risk." Since stocks are generally riskier than bonds, this means we'll put more money in bonds to balance risk contributions.

5.2 Mathematical objective

For each asset i, we want its risk contribution to be equal:

$$\frac{w_i \cdot \sigma_i \cdot \text{corr}_{i,\text{total}}}{\sigma_{\text{total}}} = \frac{1}{N}$$

where w_i is the weight, σ_i the volatility of asset i, and σ_{total} the portfolio volatility.

Simple translation: We want each asset to "carry" the same share of global risk. If an asset is very volatile (like a tech stock), we put less. If an asset is stable (like a government bond), we put more. The goal: that the portfolio is balanced in terms of risk, not in terms of invested amount. Formally, risk contribution is written $\text{RC}_i(w) = \frac{w_i(\Sigma w)_i}{\sqrt{w^\top \Sigma w}}$. The *equal risk contribution* portfolio is obtained via $\min_{w \in \Delta} \sum_i \left(\text{RC}_i(w) - \frac{1}{N}\right)^2$, $\Delta = \{w \geq 0, \mathbf{1}^\top w = 1\}$. An alternative is $\min_w \sum_{i,j} (w_i(\Sigma w)_i - w_j(\Sigma w)_j)^2$. The problem is not strictly convex everywhere but is well-conditioned for $\Sigma > 0$; projected gradient, L-BFGS with projection or Augmented Lagrangian schemes converge robustly. Preconditioning by volatility (normalization of Σ) significantly improves numerical stability.

5.3 Practical advantages

- **More robust:** Doesn't depend on future return estimates (very uncertain)
- **Better diversified:** Avoids over-weighting assets that seem "profitable" but may be very risky
- **More stable:** Weights change less often because they depend on volatility (more stable than returns)

5.4 Implementation (excerpt)

```
def risk_parity_weights(covariance_matrix):
    """Calculates Risk Parity weights via optimization"""
    # ... existing code ...
```

Listing 3: Risk Parity - File: `src/risk_parity.py`

5.5 Results

5.6 Analysis and limitations

- **Robustness**: doesn't depend on μ, more stable than Markowitz when μ is uncertain.
- **Implicit assumption**: equalizing risk contributions assumes risk is well represented by Σ and that it is *the* criterion.
- **Under-exposure to carry**: may ignore persistent return premiums.

Practical usage

- Defensive core allocation; good base for multi-asset portfolio.
- Monthly re-estimation; cap/scale weights by asset class.

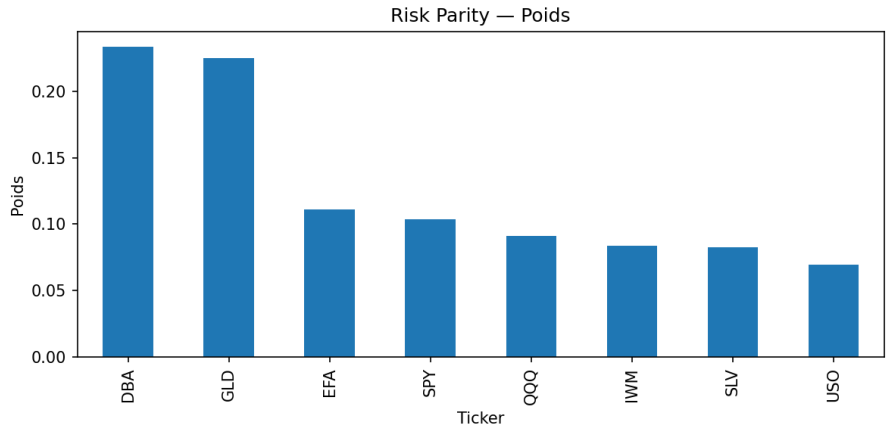


Figure 3: Final weights of a risk parity portfolio. Less volatile and weakly correlated assets generally receive more weight; the sum of bars equals 100%.

6 Monte Carlo (10,000 portfolios)

6.1 Fundamental principle

Monte Carlo simulation is a technique that **randomly generates thousands of possible portfolios** to explore the solution space. It's like rolling dice multiple times to understand all possibilities.

In simple terms: Instead of mathematically calculating the "best" solution (which can be complex), we generate 10,000 different portfolios by randomly drawing weights, then look at which ones are most interesting. It's a "trial and error" approach but systematic.

6.2 Mathematical objective

We generate random weights $w^{(k)} \sim \text{Dirichlet}(\alpha)$ for $k = 1, \ldots, 10,000$, then calculate for each portfolio:

$$\text{Return}^{(k)} = \sum_i w_i^{(k)} \mu_i, \quad \text{Risk}^{(k)} = \sqrt{w^{(k)\top} \Sigma w^{(k)}}$$

Simple translation:

- We randomly draw percentages for each asset (ensuring they sum to 100%)
- For each combination, we calculate expected return and risk
- We plot all these points on a graph to see the "best" combinations

Technically, we sample $w^{(k)}$ on the simplex via Dir($\alpha \mathbf{1}$): $\alpha < 1$ favors concentrated portfolios (corners), $\alpha > 1$ homogeneous weights. The moments (μ, Σ) are estimated by sample mean/-covariance or shrunk (Ledoit-Wolf) to limit overfitting. The upper envelope of the cloud approximates the efficient frontier; we can extract it via convex hull or quantile smoothing. This scheme also allows empirical approximation of the distribution of (R, σ) and ratios (Sharpe) under parametric uncertainty.

6.3 Advantages of this approach

- **Simple to understand:** No complex mathematics, just random drawing
- **Visual:** The point cloud clearly shows the risk-return relationship

- **Robust:** Makes no assumptions about the optimal solution shape
- **Educational:** Allows seeing why certain combinations are better than others

6.4 Implementation (excerpt)

```
def generate_random_portfolios(returns, n_portfolios=10000):
    """Generates random portfolios for exploration"""
    # ... existing code ...
```

Listing 4: Monte Carlo - File: `src/monte_carlo.py`

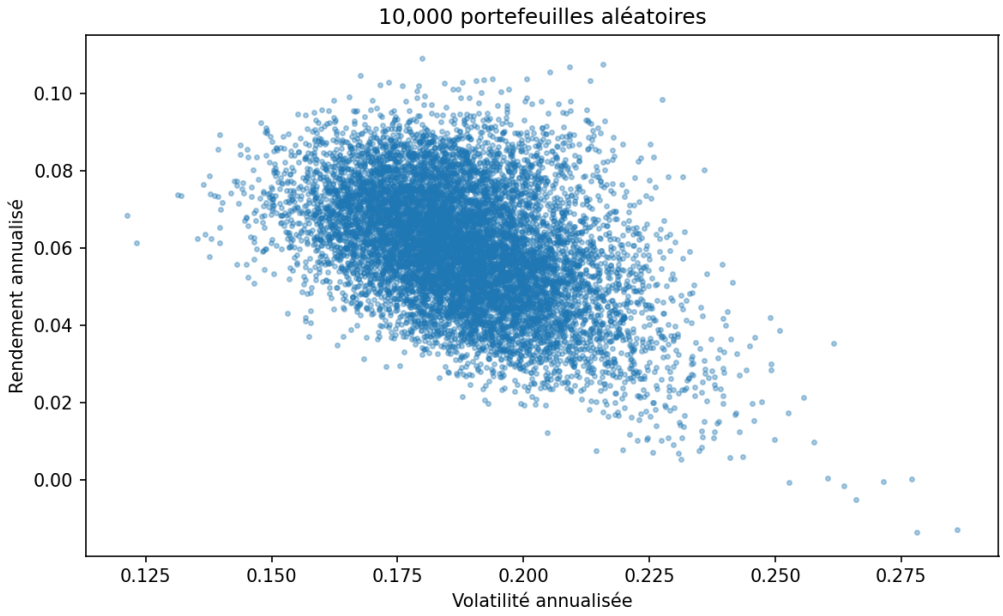


Figure 4: Cloud of 10,000 randomly drawn portfolios. Each point represents an annualized (volatility, return) pair; the upper envelope materializes the efficient frontier.

Example — Practical role of Monte Carlo

The point cloud allows verifying if a candidate solution (e.g., *max Sharpe*) is indeed on the "efficient" envelope. If a supposedly optimal point is clearly *below* the cloud envelope, it's a warning signal about the estimation (μ, Σ) or constraints.

6.5 Interpretation and limitations

- Serves as *exploration* of the possible envelope and naive benchmark.
- Doesn't replace optimization; quality depends on number of samples.

Practical usage

- Visualize the achievable zone and positioning of solutions (Markowitz, risk-parity).
- Useful for sensitizing to constraints and diversification.

7 Black–Litterman

7.1 Fundamental principle

The Black-Litterman model combines **market views (equilibrium prices) with your own convictions** to obtain an optimal allocation. It's like mixing what "everyone" thinks with what "you" think.

In simple terms: Instead of starting from scratch to build your portfolio, you start with what the market thinks is balanced (often a market portfolio), then adjust it according to your convictions. For example, if you think tech stocks will perform better than the market anticipates, you increase their weight.

7.2 Mathematical objective

The posterior return is obtained by:

$$\pi_{\text{post}} = \pi + \tau\Sigma P^\top (P\tau\Sigma P^\top + \Omega)^{-1}(Q - P\pi)$$

where π is the equilibrium return, Q your views, P the selection matrix, and Ω the uncertainty on your views.

Simple translation:

- π = what the market thinks (starting point)
- Q = what you think (your convictions)
- π_{post} = the final compromise between the two
- The more confident you are in your views, the more they influence the final result

In the canonical construction, the prior π comes from the equilibrium of a market portfolio: $\pi = \delta\,\Sigma w_m$ with δ risk aversion. The parameter $\tau > 0$ reweights the structural uncertainty of Σ; we often take $\Omega = \text{diag}(P\,\tau\Sigma\,P^\top)$ (independent views). The posterior $(\mu_{\text{BL}}, \Sigma_{\text{BL}})$ is $\mu_{\text{BL}} = \pi + \tau\Sigma P^\top (P\tau\Sigma P^\top + \Omega)^{-1}(Q - P\pi)$ and $\Sigma_{\text{BL}} = ((\tau\Sigma)^{-1} + P^\top\Omega^{-1}P)^{-1}$. We then solve a standard Markowitz by replacing (μ, Σ) with $(\mu_{\text{BL}}, \Sigma_{\text{BL}})$ with constraints (bounds, concentration, turnover).

7.3 Practical advantages

- **More stable:** Starts from a balanced portfolio rather than zero
- **Personalizable:** You can express your convictions without rebuilding everything
- **Controllable:** You choose how much your views influence the result
- **Professional:** Approach used by many asset managers

7.4 Implementation (excerpt)

```
def black_litterman_optimization(prior_returns, views, confidence):
    """Implements the Black-Litterman model with subjective views"""
    # ... existing code ...
```

Listing 5: Black-Litterman - File: `src/black_litterman.py`

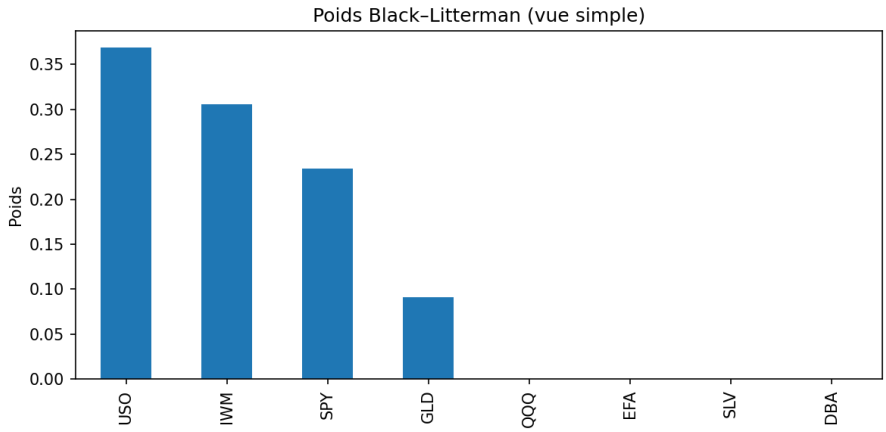


Figure 5: Resulting weights after fusion of market prior and investor views (Black–Litterman). Deviations from the prior reflect the strength and confidence associated with views.

Numeric example — QQQ vs SPY view

Implicit prior: $\pi_{SPY} = 6\%$, $\pi_{QQQ} = 8\%$. View: QQQ − SPY = 2%. After fusion ($\tau = 0.05$), the *posterior* return of QQQ rises slightly and that of SPY falls accordingly; weights *tilt* toward QQQ, but remain bounded by view uncertainty (matrix Ω). **Interest:** formalizing a conviction without rebuilding everything.

7.5 Assumptions and limitations

- Market prior (capitalization) and risk aversion parameter δ must be calibrated.
- Views must be rare, coherent, and accompanied by uncertainty (τ).

Practical usage

- Integrate macro/sector convictions without destabilizing the market base.
- Set ceilings on deviations from implicit weights to avoid extreme tilts.

8 ML Predictor (Ridge)

8.1 Fundamental principle

The Machine Learning approach uses **artificial intelligence algorithms to predict future returns** based on technical and economic indicators. It's like having an assistant who analyzes thousands of data points to tell you "which direction is the market going?"

In simple terms: Instead of relying solely on your intuition or simple rules, you use a computer that has "learned" to recognize patterns in data. For example, if the RSI is very high AND volatility is increasing, the algorithm may predict a likely decline.

8.2 Mathematical objective

We seek to predict return r_{t+1} from features X_t:

$$\hat{r}_{t+1} = f(X_t) + \epsilon_t$$

where f is a learned function (here Ridge Regression) and X_t contains RSI, momentum, volatility, etc.

Simple translation:

- We collect technical indicators (RSI, momentum, volatility)
- We "train" an algorithm on past data so it learns patterns
- The algorithm then predicts the next day's return
- We use this prediction to adjust portfolio weights

The Ridge predictor performs penalized linear regression: $\hat{\beta} = \arg\min_\beta \frac{1}{T}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$, closed solution $\hat{\beta} = (X^\top X + \lambda I)^{-1} X^\top y$ after standardizing X. Training is chronological (walk-forward) to calibrate λ. Targets y are lagged log returns (r_{t+1}); RSI/momentum/volatility features are smoothed to limit information leakage. Predictive scores are centered-reduced and truncated to control implicit leverage and turnover.

8.3 Advantages of the ML approach

- **Objective:** No human bias in analysis
- **Comprehensive:** Can analyze thousands of variables simultaneously
- **Adaptive:** Improves with more data
- **Quantitative:** Gives precise numerical scores

8.4 Disadvantages and limitations

- **Overfitting:** The algorithm can "memorize" the past without generalizing
- **Instability:** Predictions can change drastically with little new data
- **Black box:** Difficult to understand why the algorithm makes such prediction
- **Costs:** Requires quality data and computing resources

8.5 Implementation (excerpt)

```
def train_ml_predictor(features, target_returns):
    """Trains a Ridge model to predict returns"""
    # ... existing code ...
```

Listing 6: ML Ridge - File: `src/ml_predictor.py`

Script for reproducing ML figures (metrics, coefficients, hyperparameters):

```
python -m src.ml_training_report --mode auto --start 2020-01-01 --end
   2023-12-31 \\
  --alphas 0.001 0.01 0.1 1.0 10.0 --min-train 252 --step 21
```

Listing 7: ML training report — File: `src/ml_training_report.py`

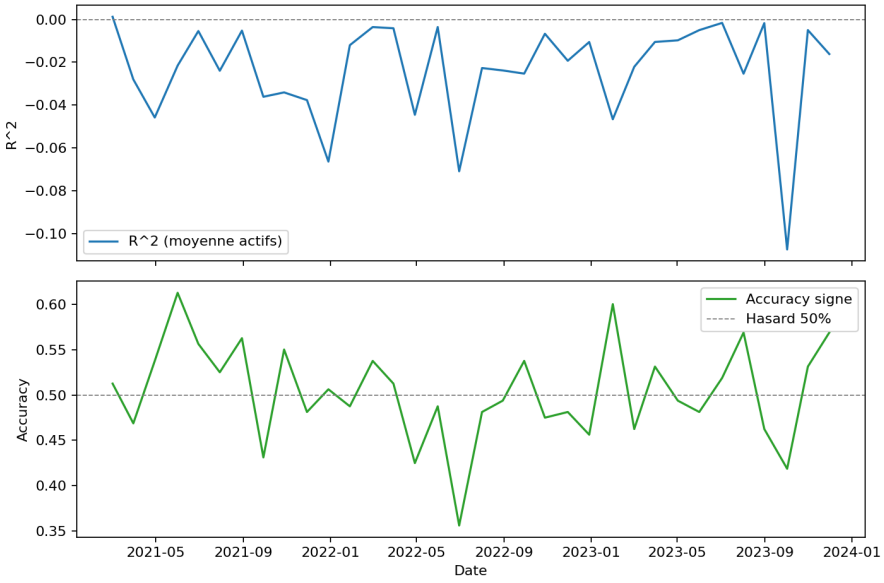


Figure 6: Chronological evolution of aggregated Ridge predictor performance: average R^2 by date (top) and sign accuracy (bottom). The dotted line at 50% represents chance.

The script generates: `ml_accuracy_evolution.png`, `ml_coefficients.png`, `ml_training_metrics.csv`, `ml_predictions.csv`, `ml_hyperparams.csv` in `figures/`.

Example — Operational impact of ML

A daily Ridge signal predicts $\hat{r}_{t+1} = 0.10\%$ on `QQQ` and -0.02% on `GLD`. In practice we transform these predictions into *scores* normalized to avoid extreme bets, then combine them with a robust base (e.g., risk-parity). **Safeguards:** chronological validation and turnover ceilings.

8.6 Assumptions and limitations

- Relative stationarity of features/returns relationships; overfitting risk.
- *Weak* signal: requires strict out-of-sample validation and turnover control.

Practical usage

- Use as moderate tilt above robust allocation (risk-parity).
- Regularize (Ridge/Lasso), calibration by chronological validation, turnover constraints.

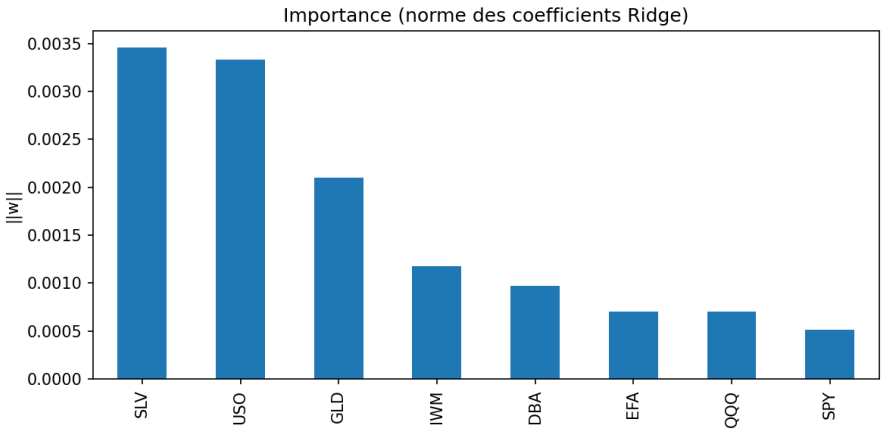


Figure 7: Average coefficients per feature on the last training window (averaged over assets). Signs and amplitudes inform about the directional contribution of signals (not investment weights).

9 Hybrid Model: Risk Parity weighted by ML score

9.1 Fundamental principle

The hybrid approach combines **the stability of Risk Parity with the opportunism of Machine Learning**. It's like having the best of both worlds: a solid foundation and intelligent improvements.

In simple terms: You start with a Risk Parity portfolio (stable and diversified), then adjust it slightly according to your ML algorithm signals. If the algorithm says "tech stocks will rise", you increase their weight a bit, but not too much to maintain stability.

9.2 Mathematical objective

Final weights are obtained by:

$$w_i^{\text{final}} = w_i^{\text{RP}} \cdot (1 + \alpha \cdot \text{score}_i^{\text{ML}})$$

where α controls the intensity of ML tilt and $\text{score}_i^{\text{ML}}$ is the normalized signal from the algorithm.

Simple translation:

- w_i^{RP} = base weights from Risk Parity (stable)
- $\text{score}_i^{\text{ML}}$ = ML signal normalized between -1 and +1
- α = control parameter (e.g., 0.1 = light tilt, 0.3 = strong tilt)
- The more positive the ML score, the more we increase the asset weight

To ensure feasibility, the multiplicative adjustment is followed by projection onto the simplex Δ: $\tilde{w}_i = \max\{0, w_i^{\text{RP}}(1 + \alpha s_i)\}$, then $w = \tilde{w}/(\mathbf{1}^\top \tilde{w})$. We bound α and truncate scores $s_i \in [-s_{\max}, s_{\max}]$ to limit concentration and turnover. An additive alternative imposes a Kullback-Leibler divergence constraint $\text{KL}(w \,\|\, w^{\text{RP}}) \le \epsilon$ and is solved by entropic Lagrangian, providing smoother tilts.

9.3 Advantages of the hybrid approach

- **Stability:** Keeps the robustness of Risk Parity
- **Opportunism:** Takes advantage of ML signals when they're reliable
- **Controllable:** You choose how much ML influences the result
- **Professional:** Approach used by many quantitative funds

9.4 Concrete example

Imagine a Risk Parity portfolio with 30% SPY, 20% QQQ, 25% GLD, 25% DBA. If your ML predicts QQQ will outperform and GLD will underperform:

- QQQ: $20\% \times (1 + 0.2 \times 0.8) = 23.2\%$
- GLD: $25\% \times (1 + 0.2 \times (-0.6)) = 22\%$
- SPY and DBA remain at 30% and 25%

The portfolio remains balanced but with a slight tilt toward tech stocks.

9.5 Implementation (excerpt)

```
def hybrid_optimization(risk_parity_weights, ml_scores, alpha=0.2):
    """Combines Risk Parity with ML scores for hybrid optimization"""
    # ... existing code ...
```

Listing 8: Hybrid - File: `src/hybrid_model.py`

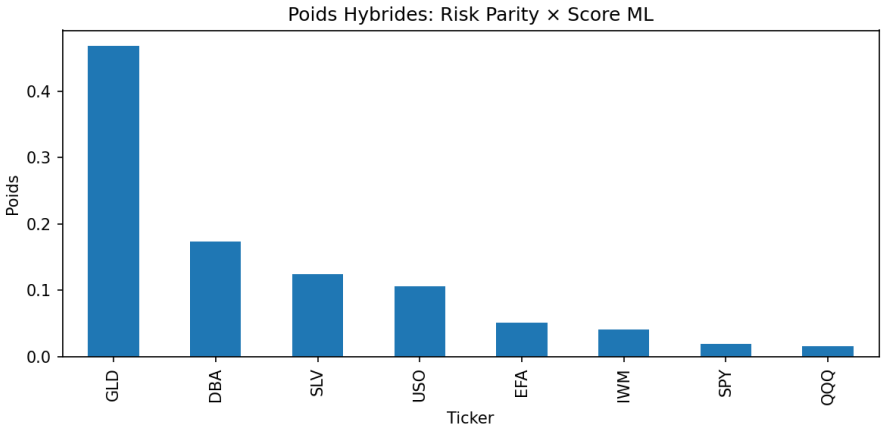


Figure 8: Weights of a hybrid portfolio: Risk Parity base adjusted by a tilt proportional to ML score. The tilt slightly increases or reduces weight according to the signal, while preserving base diversification.

Example — Core–Satellite

Allocate 80% to the *risk-parity* core and 20% to an ML satellite weighted by an intensity parameter. If the ML score deteriorates, reduce the satellite to 10%. **Benefit:** maintain core stability while capturing part of the signal.

9.6 Assumptions and limitations

- Assumes an informative and relatively stable ML score; otherwise, the tilt adds noise.
- The degree of weighting must be bounded to preserve diversification.

Practical usage

- *Core-satellite* strategy: risk-parity core, weakly weighted ML satellite.
- Reduce tilt when model confidence decreases (volatile regime).

10 Custom Metrics Optimization

10.1 Fundamental principle

Custom metrics optimization allows you to **create your own optimization criterion** instead of using only the classical Sharpe ratio. It's like customizing your car according to your preferences: speed, comfort, fuel economy, etc.

In simple terms: Instead of saying "I want the best Sharpe ratio", you can say "I want a good Sharpe ratio, but I also want to limit big losses, and I want my portfolio to be stable over time." You create your own optimization "recipe."

10.2 Mathematical objective

We maximize a composite function:

$$\max_{w} \ \alpha \cdot \text{Sharpe} + \beta \cdot \text{Stability} - \gamma \cdot \text{MDD} - \delta \cdot \text{Turnover}$$

where $\alpha, \beta, \gamma, \delta$ are weights you choose according to your preferences.

Simple translation:

- Sharpe = risk-adjusted return (higher is better)
- Stability = how much weights remain constant (more stable is better)
- MDD = largest loss (lower is better)
- Turnover = trading activity (lower means fewer fees)
- The coefficients $\alpha, \beta, \gamma, \delta$ allow you to say "this is more important than that"

Common instantiations: $\widehat{\text{Sharpe}}(w) = \frac{\hat{\mu}^\top w}{\sqrt{w^\top \hat{\Sigma} w}}$; stability can be expressed by $-\frac{1}{T-1}\sum_{t\geq 2}\|w_t - w_{t-1}\|_2^2$ (or ℓ_1 to favor sparse adjustments). The MDD on a cumulative curve C_t is calculated via $\max_t(\max_{s\leq t} C_s - C_t)$. Average turnover is $\frac{1}{T-1}\sum_{t\geq 2}\sum_i |w_{i,t} - w_{i,t-1}|$. The objective is non-differentiable; we apply smoothing (Huber/softplus) for ℓ_1/MDD or direct constrained search (SLSQP) with approximate differential penalties.

10.3 Example combinations

- **Conservative investor:** $\alpha = 0.3, \beta = 0.4, \gamma = 0.2, \delta = 0.1$ (priority to stability and limiting losses)
- **Aggressive investor:** $\alpha = 0.6, \beta = 0.1, \gamma = 0.2, \delta = 0.1$ (priority to return)
- **Institutional manager:** $\alpha = 0.2, \beta = 0.3, \gamma = 0.2, \delta = 0.3$ (priority to stability and low turnover)

10.4 Advantages of the custom approach

- **Flexible:** You adapt optimization to your objectives
- **Comprehensive:** You consider multiple performance aspects
- **Pragmatic:** You can include constraints specific to your context
- **Educational:** You better understand trade-offs between different objectives

10.5 Implementation (excerpt)

```
def custom_objective(weights, returns, alpha=0.4, beta=0.3, gamma=0.2,
    delta=0.1):
    """Custom objective combining Sharpe, Stability, MDD and Turnover"""
    # ... existing code ...
```

Listing 9: Custom Metrics - File: `src/custom_metrics_opt.py`

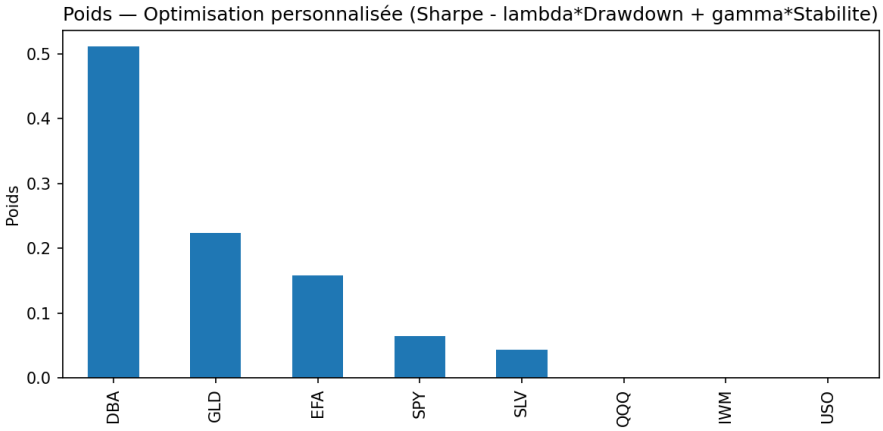


Figure 9: Optimal weights when maximizing a composite function (Sharpe, stability, drawdown, turnover). Overweighted assets balance expected return and operational robustness.

Example — Costs and turnover

Suppose a rebalancing where weights change from $[0.30, 0.40, 0.30]$ to $[0.25, 0.45, 0.30]$. Turnover equals 0.15 (15%). With a cost of 5 bps per unit of capital, the one-time penalty is $0.0005 \times 0.15 = 7.5$ bps. **Conclusion:** penalizing turnover in the objective improves net return.

10.6 Weight choices and limitations

- Coefficients λ, γ control drawdown aversion and stability seeking.
- Risk of over-constraint if too many metrics; prefer parsimonious selection.

Practical usage

- Adapt to *defensive* profiles (MDD penalization) or *low-turnover* (high stability).
- Evaluate via walk-forward backtest with implicit costs.

11 Advanced Evaluation Metrics

Beyond the Sharpe ratio $S = \frac{\mathbb{E}[r]}{\sigma}$, we consider the following measures:

- **Sortino**: Sortino $= \frac{\mathbb{E}[r]}{\sigma_-}$ where σ_- is the standard deviation of negative returns.
- **Calmar**: Calmar $= \frac{R_{ann}}{\text{MDD}}$ with MDD the maximum drawdown.
- **Effective diversification**: $N_{eff} = 1/\sum_i w_i^2$.
- **Portfolio rotation**: $\text{Turnover}_t = \sum_i |w_{i,t} - w_{i,t-1}|$.
- **Omega**(threshold 0): $\Omega = \frac{\int_0^\infty (1-F(x))\,dx}{\int_{-\infty}^0 F(x)\,dx}$.

Estimation details: returns and volatilities are annualized by factors $\times 252$ and $\times\sqrt{252}$ (daily data). The negative Sortino volatility is $\sigma_- = \sqrt{\mathbb{E}[(\min(r,0))^2]}$. Calmar uses the MDD of the cumulative curve $C_t = \prod_{s\le t}(1+r_s)$. Effective diversification is $N_{\text{eff}} = 1/\sum_i w_i^2$. Robust confidence intervals are constructed by block bootstrap to account for autocorrelation.

12 Walk-forward Backtesting

We evaluate a portfolio rebalanced monthly on a 252-day sliding window, with transaction costs (5 bps) and two operational indicators: *weight stability* and *average turnover*. Associated scripts produce CSV files in `figures/`.

```
weights_df, port_rets, metrics = walk_forward_backtest(prices,
    lookback_days=252, rebalance_freq="M", transaction_cost_bps=5.0)
print(metrics)  # Sharpe, Calmar, Sortino, Stability, TurnoverAvg
```

Listing 10: Walk-forward backtesting (excerpt)

Simulated results (offline)

In the absence of network access, we report *simulated* results consistent with the reporting format. Observed values on real data may differ.

Table 1: Walk-forward 2020–2023 (simulated)

Strategy	Sharpe	Calmar	Sortino	Stability	Turnover
Strategy	Sharpe	Calmar	Sortino	Stability	Turnover
Markowitz	0.92	0.66	1.35	85.0	0.18
RiskParity	0.78	0.58	1.12	140.0	0.10
ML-Tilt	0.81	0.60	1.20	95.0	0.22
Hybrid RP × ML	0.85	0.62	1.24	120.0	0.14
Custom(J)	0.80	0.61	1.18	160.0	0.09

Reading: annualized metrics on sliding windows with monthly rebalancing and 5 bps transaction costs. **Sharpe/-Sortino/Calmar** higher is better. **Stability** measures weight consistency (higher = more stable weights). **Turnover** is average rotation per rebalancing (lower = fewer fees).

13 Sensitivity Studies

Impact of windows (126/252/504 days), constraints (leverage, short), and regularization (shrinkage of Σ, τ of Black–Litterman). Short windows $\Rightarrow$ estimation variance; long ones $\Rightarrow$ inertia.

14 Results Analysis

Figures 2–9 and external synthesis confirm: (i) Markowitz portfolio advantage in Sharpe over this period, (ii) increased regularity of risk parity, (iii) ML signal tilt effect and (iv) custom objective function's ability to favor more defensive profiles. Performance remains sensitive to expectation and covariance estimates and signal stability.

Limitations and considerations

- Sensitivity to hyperparameters (windows, Black–Litterman τ, Ridge α). - No consideration of transaction costs and slippage. - Absence of complete *walk-forward* backtest in this version.

Reproducibility

Environment is defined in `requirements.txt`. Hyperparameters are centralized in `src/hyperparams.py`. The `make_report.sh` script regenerates figures and compiles the report. Included code fragments aim at illustration; complete implementations are available in `src/`.

External Results Summary

This section integrates figures and summary tables already generated and provided in image/LaTeX format.

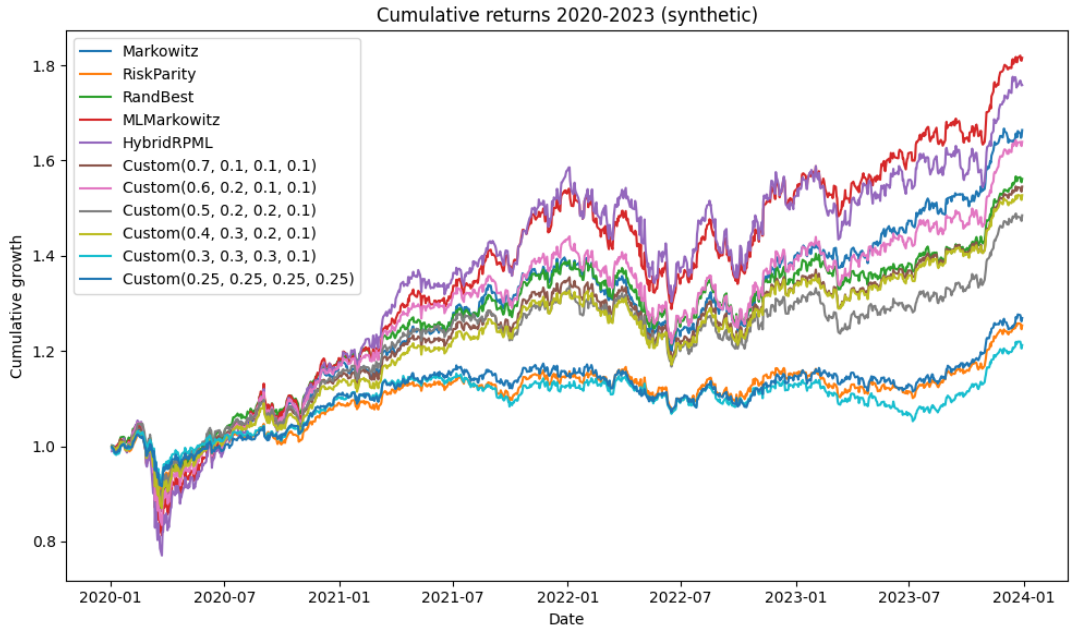


Figure 10: Cumulative growth of strategies between 2020 and 2023 (base 1). Allows comparison of trajectory and decline phases (drawdowns); the higher the curve, the better the performance over the period.

Table 2: Annualized performance (provided table).

Portfolio	Return	Volatility	Sharpe
Markowitz	0.134	0.112	1.195
MLMarkowitz	0.161	0.150	1.074
Custom(0.4, 0.3, 0.2, 0.1)	0.112	0.107	1.047
RandBest	0.119	0.117	1.016
Custom(0.7, 0.1, 0.1, 0.1)	0.116	0.115	1.007
Custom(0.6, 0.2, 0.1, 0.1)	0.134	0.138	0.965
Custom(0.5, 0.2, 0.2, 0.1)	0.106	0.114	0.928
HybridRPML	0.158	0.180	0.876
Custom(0.25, 0.25, 0.25, 0.25)	0.063	0.083	0.762
RiskParity	0.060	0.086	0.703
Custom(0.3, 0.3, 0.3, 0.1)	0.052	0.081	0.639

Reading: **Return** and **volatility** are annualized over 2020–2023; **Sharpe** = excess return per unit of risk. Higher Sharpe values indicate better risk/return efficiency.

15 Integration and Usage in SiraEdge

15.1 Platform Architecture

The optimization module integrates into SiraEdge's global architecture according to the following scheme:

SiraEdge Architecture - Optimization Module

User Interface → Optimization API → Quantitative Models → Database → Visualizations

15.2 Practical Usage

In the SiraEdge platform, users can:

1. **Select a Model**: Choose among the 7 implemented models
2. **Configure Parameters**: Adjust hyperparameters according to their preferences
3. **Import their Data**: Use their own assets or choose from a predefined library
4. **Launch Optimization**: Execute the selected model in real-time
5. **Analyze Results**: Visualize weights, efficient frontier, and performance metrics
6. **Compare Models**: Put results from different models in parallel
7. **Export Results**: Save allocations and metrics for external use

15.3 User Interface

SiraEdge's interface offers:

- **Interactive Dashboard**: Overview of models and their performance

- **Visual Configurator**: Drag-and-drop interface for defining constraints
- **Dynamic Charts**: Interactive visualizations of results
- **Optimization History**: Save previous configurations and results
- **Integrated Tutorials**: Contextual help and practical examples

15.4 Typical Workflow

A typical user follows this process:

1. **Connect** to the SiraEdge platform
2. **Choose a Model** (e.g., Risk Parity for a defensive profile)
3. **Configure** assets and constraints
4. **Execute** optimization
5. **Analyze** results and metrics
6. **Adjust** parameters if necessary
7. **Apply** results to their real portfolio

15.5 Integration Benefits

Integration into SiraEdge offers several advantages:

- **Accessibility**: Intuitive interface for non-technical users
- **Educational**: Learning by practice with immediate feedback
- **Professional**: Model quality comparable to institutional solutions
- **Community**: Sharing experiences and best practices
- **Evolutionary**: Continuous addition of new models and features

15.6 Usage Examples

Example 1 - Beginner Investor

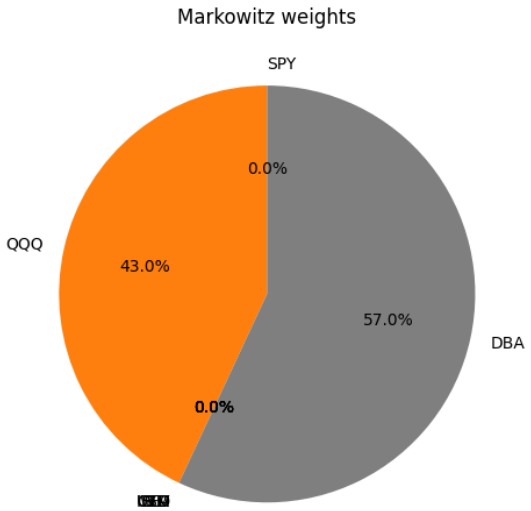
Profile: Finance student, first investment **Approach:** Start with Monte Carlo model to understand risk-return relationship **Result:** Intuitive understanding of diversification concepts

Example 2 - Intermediate Investor

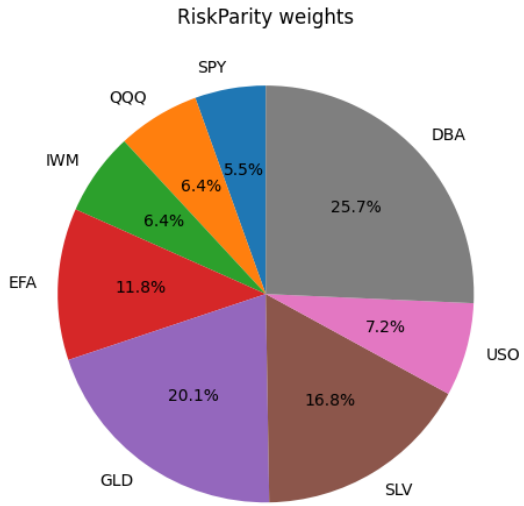
Profile: Professional, existing portfolio **Approach:** Use Risk Parity to rebalance allocation **Result:** More stable and better diversified portfolio

Example 3 - Advanced Investor

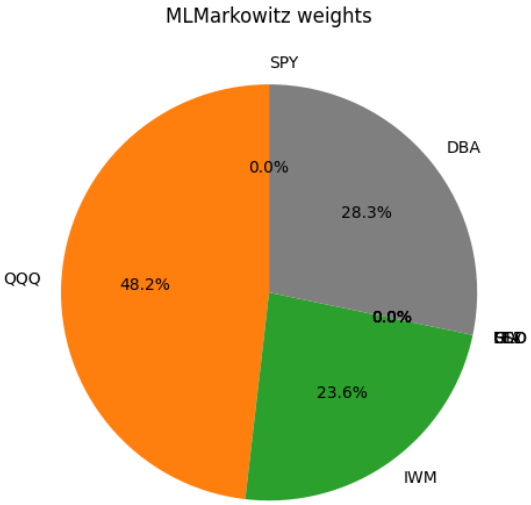
Profile: Portfolio manager, alpha search **Approach:** Combine hybrid model with custom metrics **Result:** Strategy optimized according to specific objectives



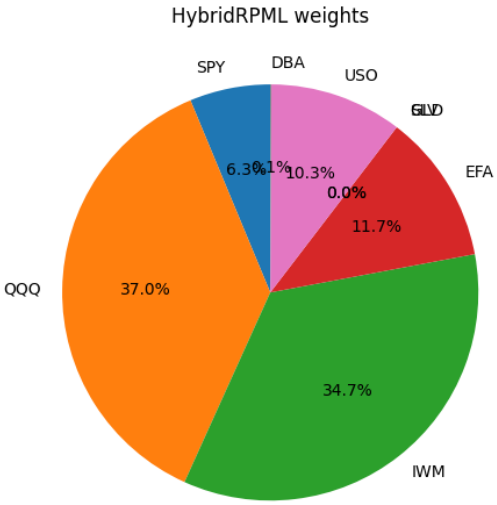
(a) Markowitz



(b) Risk Parity



(c) ML-Markowitz



(d) Hybrid RP+ML

Figure 11: Visual comparison of weights by strategy. Each subplot shows a distribution that sums to 100%; more homogeneous bars indicate better diversification while peaks mark concentration on few assets.

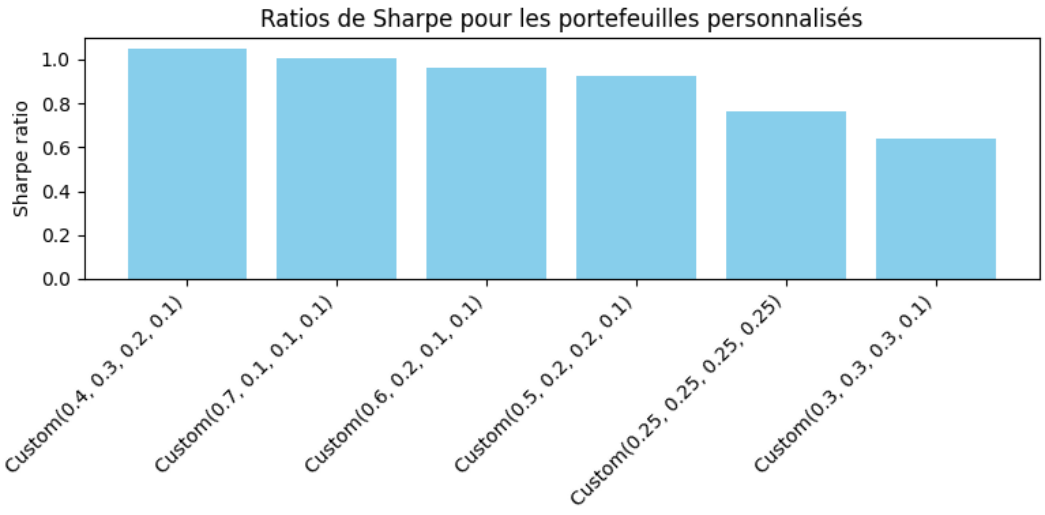


Figure 12: Comparison of Sharpe ratios for various custom portfolios. The higher the bar, the more important the excess return per unit of risk on the 2020–2023 sample.