# Markowitz vs RL Portfolio Optimization

Andy Au

April 30, 2025

# Goal of Project

- Compare classic Markowitz Portfolio Optimization with RL (Reinforcement Learning) based optimization
- Evaluate based on financial metrics: Sharpe ratio, return, volatility

# Data

- We selected just 10 ETFs: SPY, QQQ, IWM, EFA, EEM, VNQ, TLT, IEF, GLD, USO
- Data sourced from Yahoo Finance (yfinance)
- Time Period: 2018.01.01 - 2024.01.01
- Frequency: Daily adjusted close prices

# ETF Price Visualization

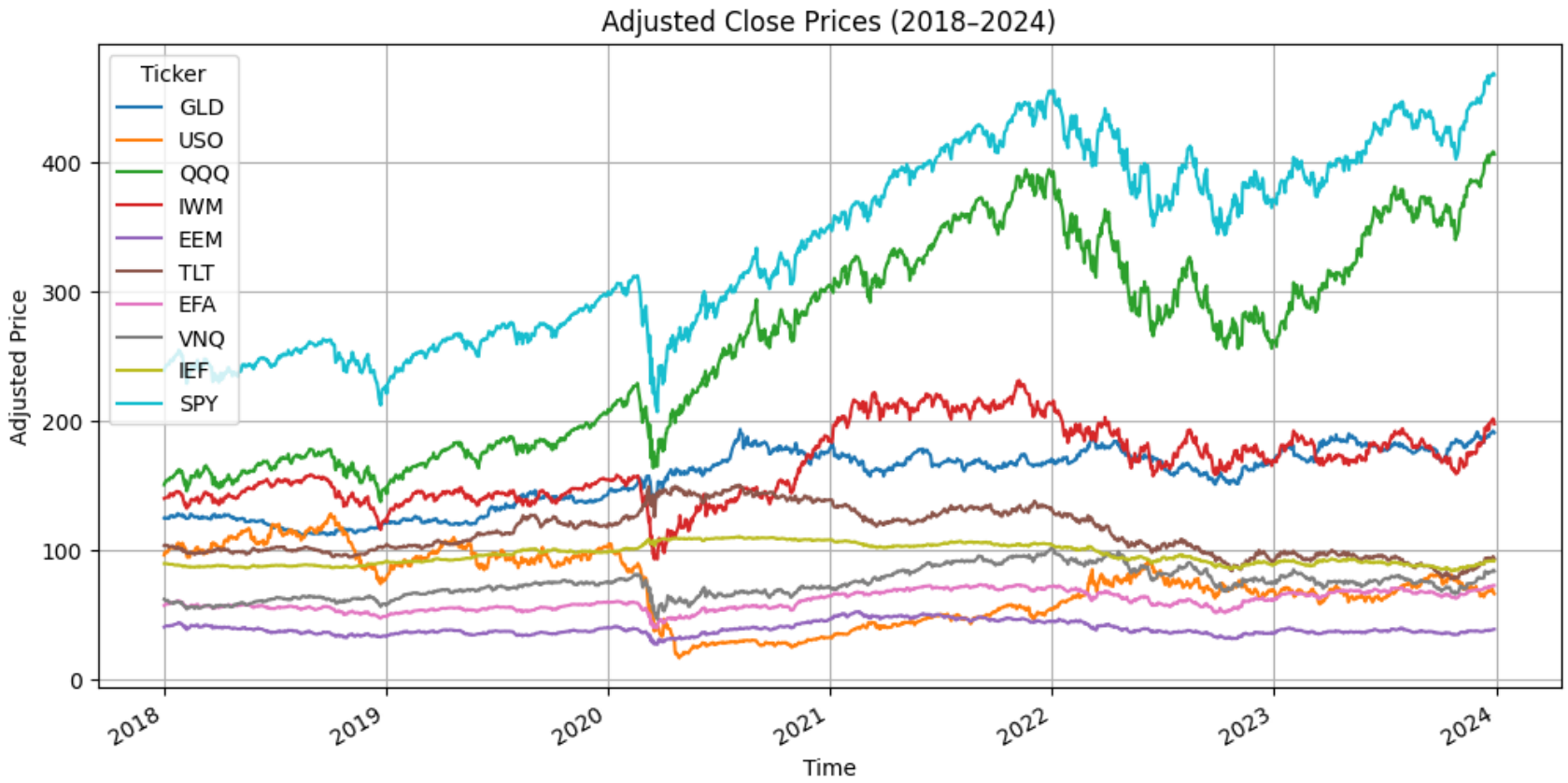


Figure 1: Adjusted Prices of our 10 ETFs

# Baseline: Naive Portfolio

- For comparison, we construct a naive portfolio, with equal weight allocation across all ETFs
- We will return to this later during comparison

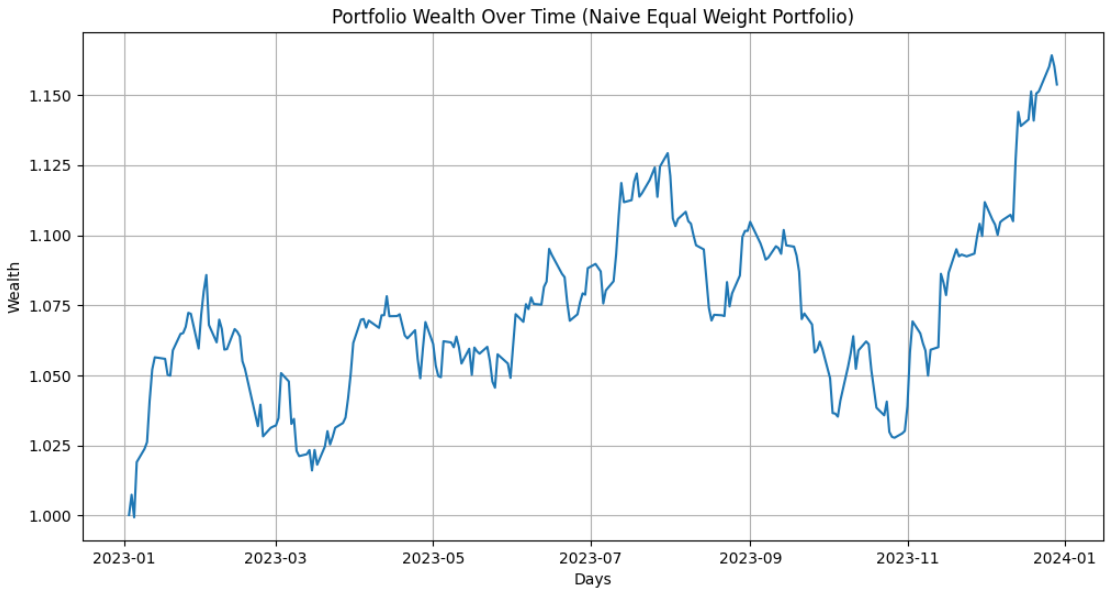


Figure 2: Naive Portfolio Wealth from 2018-2024

# Markowitz Optimization

- Construct a portfolio that balances expected return and volatility
- Return is the weighted average of asset returns
- Risk is the portfolio variance, based on asset correlations
- Optimize to maximize the Sharpe Ratio:

$$\text{Sharpe} = \frac{w^T \mu - r_f}{\sqrt{w^T \Sigma w}}$$

- Subject to:
  - $\sum w_i = 1$ (full investment)
  - $w_i \geq 0$ (no short selling)

# Markowitz: Walkthrough

1. Download daily ETF price data from 2018 - 2023
2. Calculate daily returns from adjusted close prices
3. Compute:
   - Mean returns vector $\mu$
   - Covariance matrix $\Sigma$
4. Set up an optimization problem to maximize Sharpe Ratio
5. Solve for optimal weights using `scipy.optimize.minimize`
6. Evaluate the resulting portfolio on 2023 - 2024 out-of-sample data

# Markowitz: Implementation

- **Objective:** Maximize Sharpe Ratio (minimize negative Sharpe)
- **Constraints:**
  - Weights must sum to 1
  - No short selling (weights $\geq 0$)
- **Method:**
  - Use `SLSQP` solver in `scipy.optimize.minimize`
  - Initial guess: equal weight allocation across all ETFs
  - Input mean returns and covariance matrix from training data
- **Outputs:**
  - Optimal portfolio weights
  - Annualized return, volatility, and Sharpe Ratio
  - (Portfolio visualization)

# Correlation Matrix Visualization

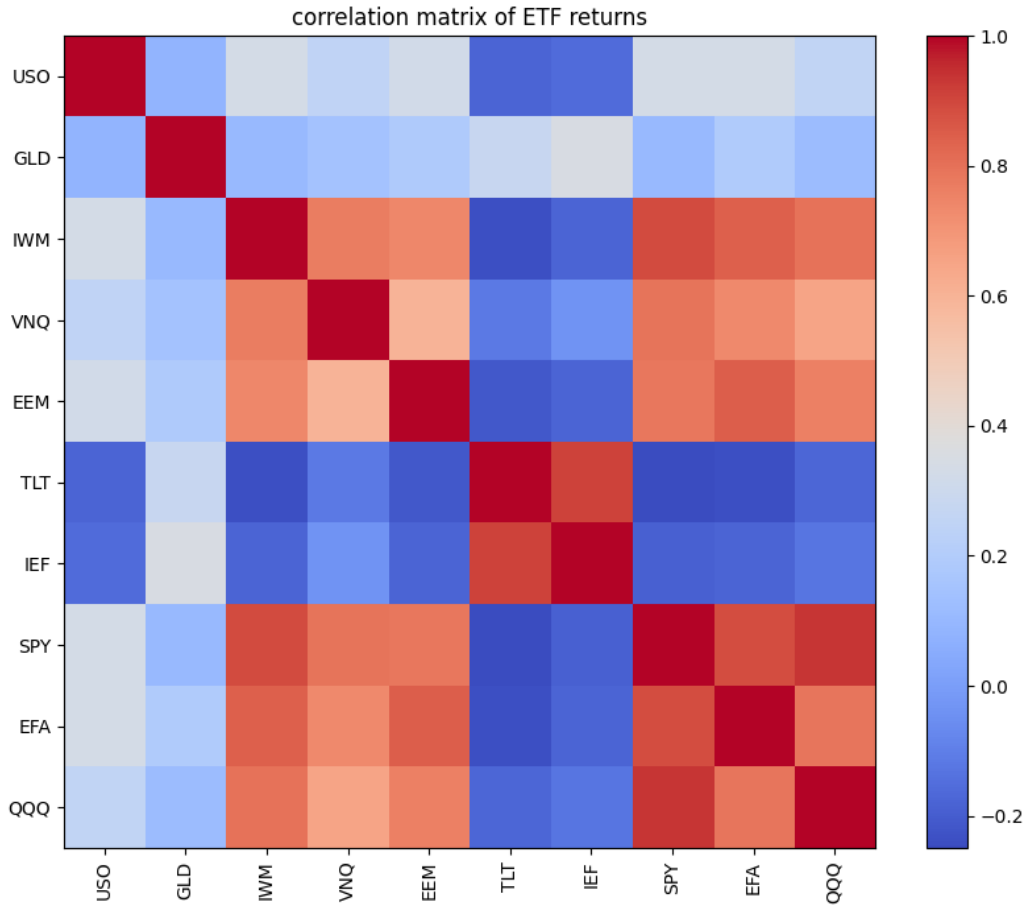


Figure 3: Correlation Matrix of ETF returns (train data)

# Markowitz: Results

- **Annualized Return:** 25.37%
- **Annualized Volatility:** 10.76%
- **Sharpe Ratio:** 2.36
- Portfolio heavily weighted in QQQ and USO
- Overall solid performance during 2023-2024

# Markowitz: Portfolio Visualization

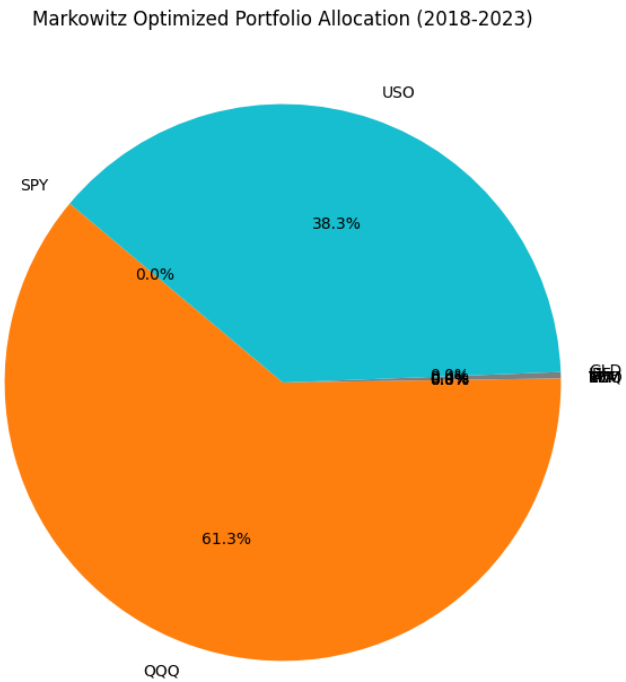


Figure 4: Portfolio Weights


Figure 5: Wealth Growth in 2023

# Markowitz Discussion

- Optimized portfolio achieved high return and Sharpe ratio on 2023 out-of-sample data
- Heavy concentration in few ETFs
- Relies heavily on accurate estimation of expected returns and covariances
- Assumes static market behavior — cannot adapt dynamically to shocks
- May suffer from overfitting to historical period (estimation risk)

# Introduction to RL (Reinforcement Learning)

- RL has an agent which learns by interacting with an environment to maximize cumulative rewards
  Now what does that mean?
- In portfolio management, the agent reallocates investments dynamically based on market data
- Unlike static optimization (e.g. Markowitz), RL agents adapt to changing conditions over time
- The idea is to train an agent to learn optimal portfolio strategies from historical ETF returns

# RL Approach Intuition

- **Goal:** Maximize risk-adjusted returns by adjusting portfolio weights daily
- **Observation:** Past 30 days of ETF returns (rolling window)
- **Action:** Assign a new portfolio allocation across the 10 ETFs
- **Reward:** Higher portfolio returns with lower volatility (Sharpe-like reward)
- **Adaptivity:** Agent learns to react to market trends and volatility shifts

# RL Formulation and Portfolio Context

Modeling our RL problem

- $\mathcal{S}$: State space – past 30 days of ETF returns
- $\mathcal{A}$: Action space – portfolio weight vector across 10 ETFs
- $r(s, a)$: Reward – log return minus volatility penalty

**Agent's Objective:**

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$$

**Policy** $\pi$ maps states to actions, learned to maximize $J(\pi)$

# Deep RL with PPO: Training Workflow

**Policy Class:** PPO (Proximal Policy Optimization) with Actor-Critic structure

- **Actor:** selects action (portfolio weights) based on state
- **Critic:** estimates value of state to guide learning

**Training Loop:**

1. Get state $s_t$ (30-day rolling return window)
2. Agent selects action $a_t$ (weights)
3. Environment returns reward $r_t$ and next state $s_{t+1}$
4. Update network based on PPO loss

**Reward Function:**

$$r_t = \log(1 + R_t^{\text{portfolio}}) - \lambda \cdot \text{VolatilityPenalty}$$

**Goal:** Maximize return while penalizing high volatility

# RL Implementation

1. Download daily ETF data from 2018 - 2023
2. Define a custom environment with a rolling 30-day observation window
3. Train a PPO agent for 500,000 timesteps
4. Evaluate the agent's performance on 2023-2024 out-of-sample data
5. Measure performance: annualized return, volatility, and Sharpe Ratio
6. Visualize wealth growth and dynamic portfolio weights over time

# RL: Results

- **Average Annualized Return:** 18.99%
- **Average Annualized Volatility:** 10.52%
- **Average Sharpe Ratio:** 1.81

# RL: Portfolio Visualization (1)

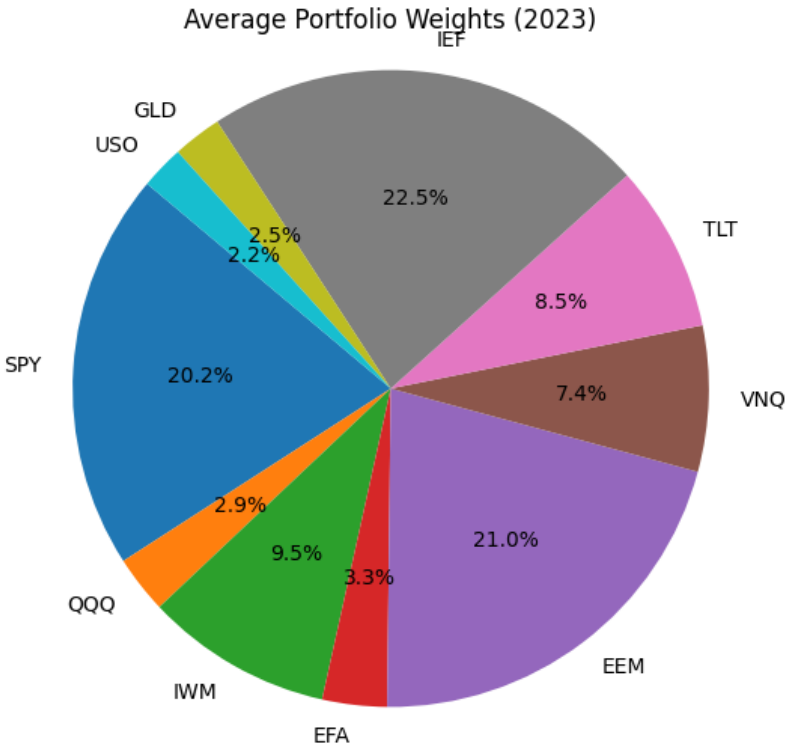


Figure 6: Averaged Portfolio Weights

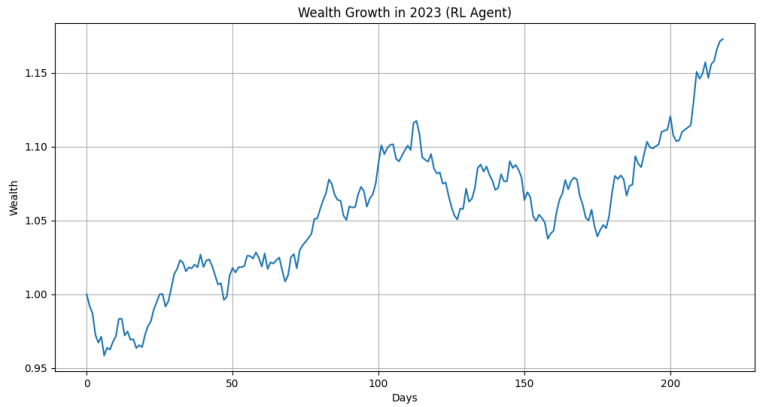


Figure 7: Wealth Growth in 2023

# RL: Portfolio Visualization (2)

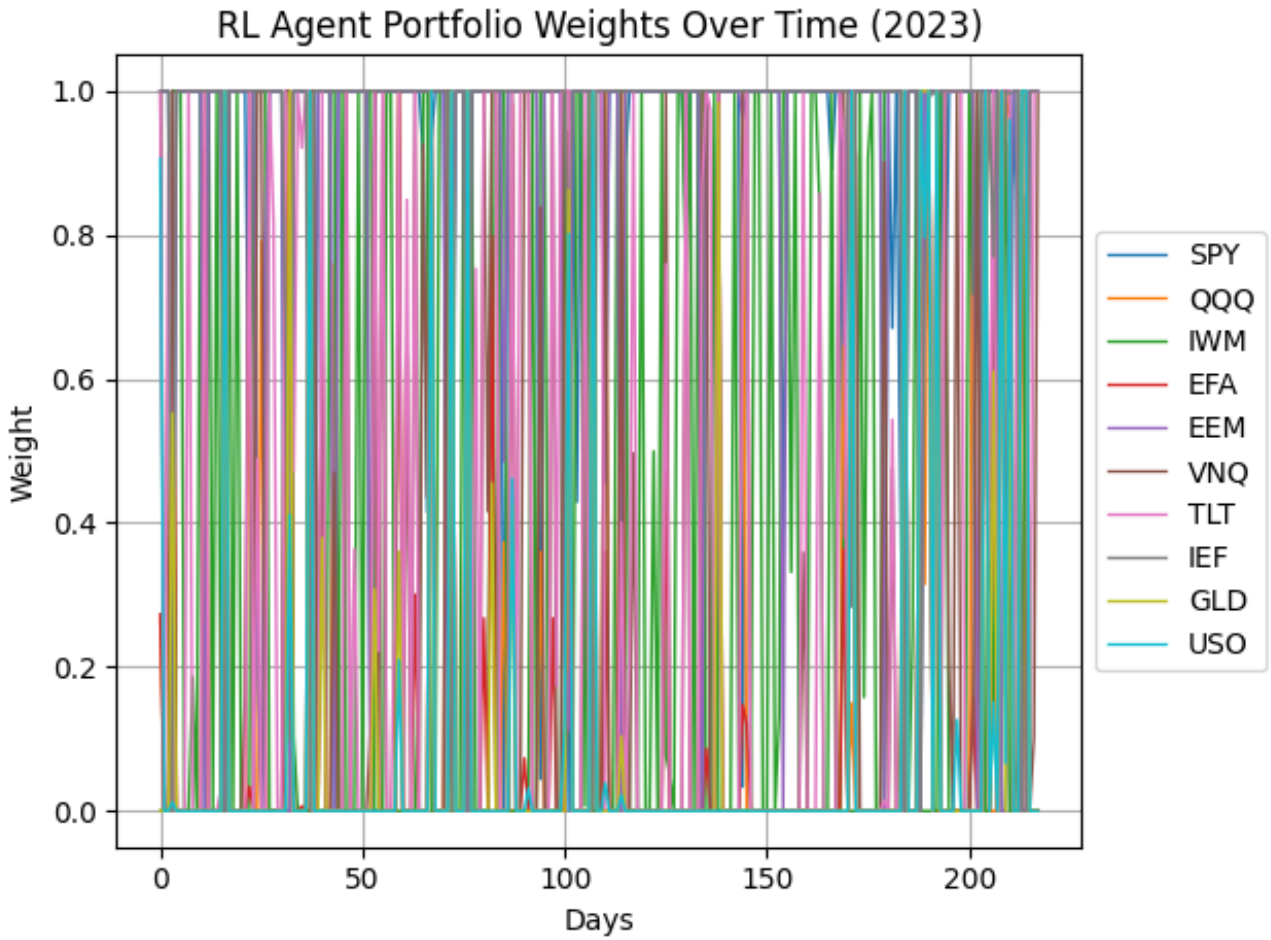


Figure 8: Selected Weights Over Time

# RL Discussion

- **Strengths:**
  - Can dynamically adapt portfolio allocations based on market trends and volatility
  - Does not rely on static assumptions about mean returns or covariance matrices
  - Can be further optimized
    (Meanwhile Markowitz is a "one and done" solution)
- **Weaknesses:**
  - Training is computationally intensive and sample inefficient
  - Results are more volatile and harder to interpret than static models
  - Requires careful environment design, reward shaping, and hyperparameter tuning

# Comparison of Portfolios

- Summary of metrics across Naive, Markowitz, and RL portfolios:

| **Portfolio** | **Annualized Return** | **Volatility** | **Sharpe Ratio** |
|---|---|---|---|
| Naive (Equal Weight) | 15.00% | 10.70% | 1.40 |
| Markowitz | 25.37% | 10.76% | 2.36 |
| RL (PPO Agent) | 18.99% | 10.52% | 1.81 |

- Markowitz achieved the highest Sharpe ratio, indicating the best risk-adjusted performance
- RL achieved higher return and Sharpe ratio than the naive portfolio, but lower than Markowitz
- Naive portfolio had poor risk-adjusted returns despite diversification

# Other Notes

**Volatility Penalty Adjustment:**

- Early versions of the RL agent achieved extreme performance - $\sim$ 1200% returns with $\sim$ 300% annualized volatility
- Although Sharpe ratios were numerically high, such portfolios would be considered unrealistic or unacceptable in real-world settings (?)
- A volatility penalty was introduced to encourage more stable, feasible portfolio strategies

# Future Directions

- Incorporating transaction costs and turnover penalties into the environment
- Further adjusting PPO parameters such as learning rate, clip range, entropy coefficient, and batch size to optimize agent performance
- Exploring more advanced models such as LSTM-based or attention-based policies
- Improving reward shaping to better balance risk
- Expanding observations to include additional features beyond historical returns (e.g., volatility, macroeconomic indicators)

# Conclusion

- Markowitz portfolio optimization achieved the best risk-adjusted performance (highest Sharpe)
- RL-based portfolio outperformed the naive equal-weight portfolio in both return and Sharpe ratio
- RL offers dynamic adaptability and flexibility to changing markets, unlike the static Markowitz approach
- However, RL methods introduce greater complexity, training costs, and higher result variance