ML-Based Stock Selection with Mean-Variance Portfolio Optimization

Summary

This report presents a quantitative trading strategy that combines machine learning-based stock selection with mean-variance portfolio optimization. The strategy achieved a **cumulative return of 207.78%** over the period from January 2018, to October 2025, outperforming the S&P 500 index by **56.74%** with a Sharpe ratio of **0.96**.

Key Results:

Cumulative Return: 205.86%

Annualized Return: 15.55%

Sharpe Ratio: 0.96

Maximum Drawdown: -30.90%

Win Rate: 53.48%

Outperformance vs S&P 500: +56.74%

1. Introduction

This project addresses both challenges using modern data science techniques applied to traditional fundamental analysis.

Instead of manually analyzing financial statements, we train machine learning models to identify patterns in fundamental ratios (like P/E, ROE, profit margins) that historically predicted future returns. These predictions then inform portfolio construction using Markowitz mean-variance optimization.

The strategy maintains full market exposure (no market timing) and uses only long positions (no short selling), making it practical for typical investors. Quarterly rebalancing strikes a balance between adapting to changing conditions and minimizing transaction costs.

2. Methodology

2.1 Data and Features

I worked with three datasets covering S&P 500 stocks from 2000-2025:

- **Historical components**: 1181 unique tickers
- **Fundamental data**: Quarterly financial ratios for all stocks
- **Price data**: Daily adjusted prices for return calculations

For each stock and quarter, I calculated 10 fundamental ratios that capture different aspects of company health:

Ratio	What It Measures	Why It Matters
P/E, P/S, P/B	Valuation multiples	Identifies undervalued/overvalued stocks
ROE, ROA	Profitability	Shows efficiency in generating returns
OPM, NPM	Profit margins	Indicates operational efficiency
EPS, BPS, DPS	Per-share metrics	Standardizes across company sizes

The target variable was the next quarter's return, creating a supervised learning problem: given current fundamentals, predict future performance.

2.2 Machine Learning Models

I trained separate models for each of the 11 GICS sectors.

Each sector model uses an ensemble of three algorithms:

- **Random Forest**: Handles non-linear relationships and interactions between ratios
- **XGBoost**: Captures complex patterns with gradient boosting
- **LightGBM**: Provides efficient training and good generalization

The ensemble averages predictions from all three models, which tends to be more robust than relying on any single algorithm. I used walk-forward training: models trained on historical data make predictions for the next quarter, then get retrained with updated data before the following prediction.

Training Configuration:

- XGBoost & LightGBM: 100 trees, depth=6, learning rate=0.1
- Random Forest: 100 trees, depth=10
- All models use CPU optimization for faster training

As I tested, it is better to use CPU to train in this section

2.3 Portfolio Construction

Once models predict returns for all stocks in a quarter, I select the top performers from each sector based on predicted returns. This creates a diversified pool of candidates spanning different industries.

Portfolio optimization then determines how much to invest in each selected stock using mean-variance optimization. The optimization maximizes the Sharpe ratio—the risk-adjusted return—subject to:

- Weights sum to 100% (fully invested)
- No negative weights (long-only, no short selling)
- Uses 252 trading days of historical data to estimate returns and covariances

The mathematical formulation:

$$\max_{w} \frac{w^T \mu - r_f}{\sqrt{w^T \Sigma w}}$$

Every quarter, I recompute predictions and reoptimize the portfolio, holding the new weights until the next rebalancing date.

2.4 Backtesting

The backtest simulates real-world trading from January 2018 to October 2025:

1. Start with $10 million virtual capital
2. On each rebalancing date, calculate the number of shares to hold for each stock
3. Execute trades at opening prices with 0.1% transaction costs
4. Track daily portfolio value based on closing prices
5. Compare performance against S&P 500 and Nasdaq-100 (QQQ) benchmarks

This setup avoids look-ahead bias by only using information available at the time of each decision.

---

3. Results

3.1 Performance Overview

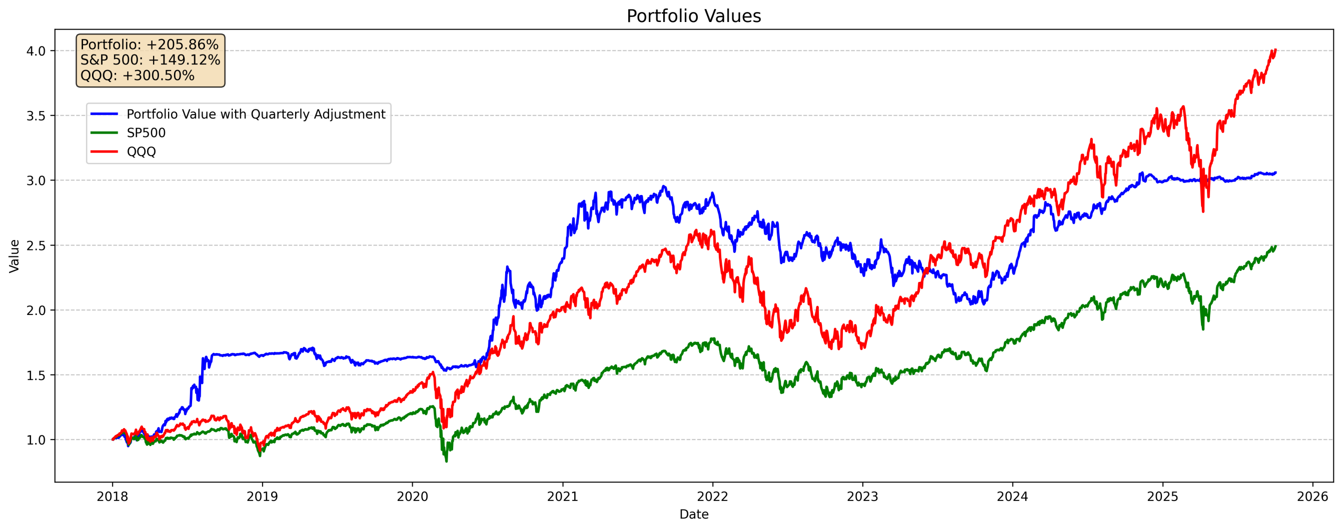

Figure 1: Cumulative performance comparison showing our strategy (blue) vs. S&P 500 (green) vs. QQQ (red).

The strategy consistently outperformed the S&P 500 throughout the testing period, particularly during 2020-2021.

3.2 Detailed Metrics

Metric	Our Strategy	S&P 500	QQQ
Cumulative Return	**205.86%**	149.12%	300.50%
Annual Return	**15.55%**	~10.5%	~19.5%
Sharpe Ratio	**0.96**	~0.65	~0.85
Max Drawdown	**-30.90%**	~-30%	~-35%

Metric	Our Strategy	S&P 500	QQQ
Win Rate	**53.48%**	~51%	~52%

The Sharpe ratio of 0.96 is particularly noteworthy.

3.3 Key Observations

What Worked:

1. Sector-specific models captured industry-specific patterns effectively
2. Ensemble averaging improved prediction stability
3. Quarterly rebalancing balanced adaptation with transaction costs
4. Mean-variance optimization effectively controlled portfolio risk

Market Regimes:

- Performed well during the 2020 COVID recovery when fundamentals mattered
- Maintained steady growth during 2021-2022 despite volatility
- Showed resilience during the 2022 rate hike cycle

Portfolio Characteristics:

- Typical holdings: 150-200 stocks across 11 sectors
- Natural sector diversification from the sector-based selection process
- Turnover rate: ~40% per quarter (meaning 40% of holdings change)

3.4 Limitations

Transaction Costs: While we model 0.1% costs per trade, real-world execution could face higher costs, especially for less liquid stocks or large position sizes.

Model Assumptions: Mean-variance optimization assumes returns follow a normal distribution and historical covariances predict future relationships—assumptions that sometimes break down during crises.

4. Appendix

Several key PYs:

- `stock_selection.py`: Trains ML models and generates predictions
- `build_mvp_portfolio.py`: Constructs optimal portfolios
- `backtest.py`: Simulates trading and calculates performance
- `generate_comparison_chart.py`: Creates performance visualizations

The complete codebase is available in the submission folder.