Alpha Allocator: Portfolio Optimization via Sector-Relative Machine Learning

Vlad Sandrovschi
vlad.sandrovschi@unil.ch

January 11, 2026

Abstract

Constructing an equity portfolio that consistently outperforms the S&P 500 benchmark is a persistent challenge in quantitative finance, often hampered by low signal-to-noise ratios and broad market regime shifts. Traditional absolute return predictions frequently fail by merely capturing market beta rather than true alpha. This project addresses this limitation by developing a machine learning pipeline designed to identify high-conviction "Buy" signals through a novel sector-relative performance ranking approach.

Leveraging a dataset of approximately 320 US equities across 11 GICS sectors spanning 2015–2025, I engineered a set of "Smart Beta" features focused on momentum, relative strength, and trend distance. A **Histogram Gradient Boosting Classifier** was trained to predict whether a stock would outperform its specific sector median over a 3-month horizon, effectively neutralizing broad market movements. By dynamically optimizing the decision threshold to target the top 10% of confidence scores, the model achieved a **54.6% Win Rate** on the out-of-sample test set (2024–2025), significantly exceeding the random baseline.

These predictive signals drive a dynamic portfolio simulation engine implementing a "Zero-Tolerance" strategy to high-conviction assets while completely divesting from low-conviction holdings. The resulting Monte Carlo simulations demonstrate that the optimized portfolio exhibits a superior projected risk-return profile compared to the passive benchmark under 200 distinct market scenarios.

Contents

1 Introduction

1.1 Motivation

As a member of a student investment association at HEC Lausanne, we actively manage a fund of our own money through equity trades. We pitch and identify likely stocks to outperform the S&P 500. It is a challenging task to find such companies given the size of the market. Stocks within the same sector often diverge significantly based on idiosyncratic factors. Traditional screening methods (e.g., P/E ratio cutoffs) are static and fail to capture non-linear relationships in market data.

This project aims to bridge the gap between Data Science and Asset Management by building a Machine Learning (ML) classifier that tries to predict if a certain stock beats the benchmark over a 3-month period. It is not merely a stock predictor but a tool that integrates machine learning predictions for stock picking with an asset allocation dashboard (Black-Litterman tilting) to create an investment strategy.

1.2 Problem Statement

Predicting raw stock returns is notoriously difficult due to low signal-to-noise ratios. A model trained to predict if a stock beats the S&P 500 often fails during market downturns (where no stock beats the benchmark) or tech bubbles (where only Tech beats the benchmark). This creates a bias where the model simply learns to be "Long Tech" or "Short Energy."

1.3 Objectives

- **Data Engineering:** Construct a robust pipeline handling $\sim$320 stocks over 10 years, adjusting for sector-specific nuances.
- **Model Optimization:** Classification in a sector-relative ranking, targeting the top decile of performers.
- **Simulation:** Validate the strategy using a Monte Carlo simulation that compares the ML-enhanced portfolio against a standard passive benchmark.
- **Dashboard:** Visualize the results in an interactive dashboard for user use.

2 Research Question & Literature

Research Question: *Can a ML model, trained on sector-relative technical indicators, identify stocks that statistically outperform a passive market-cap-weighted benchmark?*

2.1 Related Work

- **Fama & French (1993):** Established that specific "factors" (like size and value) drive returns. This concept was used as a non-linear ML models to find dynamic price-action factors rather than just static fundamental ratios.
- **Gu, Kelly, & Xiu (2020):** Demonstrated that machine learning models, specifically decision trees and neural networks, outperform standard econometric regressions in asset pricing. This research validates my choice of tree-based models (Gradient Boosting) over simple linear regressions.
- **Relative Strength (Levy, 1967):** The concept that strong assets tend to outperform weak assets over time. Calculating relative strength *within* sectors (e.g., comparing Apple to Microsoft) rather than across the whole market to isolate alpha was integrated.

3 Methodology

3.1 Data Acquisition

The tool `yfinance` API was integrated to ingest daily adjusted closing prices for a diversified universe of 328 US equities and 8 Macro ETFs (Bonds, Commodities, Real Estate) from January 1, 2015, to December 30, 2025.

To ensure realistic portfolio construction, stocks were segmented into 11 sectors based on the **Global Industry Classification Standard (GICS)** (e.g., Tech, Financials, Healthcare). This segmentation is critical for the feature engineering phase.

3.2 Feature Engineering

A "Smart Beta" feature set was created to capture price dynamics relative to the market:

1. **Momentum:** 1-month (`Ret_1M`) and 3-month (`Ret_3M`) rolling returns.
2. **Relative Strength Index (RSI):** A 14-day technical indicator that measures the speed and change of price movements. It ranges from 0 to 100 and helps identify overbought or oversold conditions.
3. **Trend Distance (`Trend_SMA`):** The percentage distance of the current price from its 50-day Simple Moving Average. Positive values indicate an uptrend.
4. **Market Relative Strength (`Rel_Str_3M`):** The stock's performance compared to the S&P 500 performance over the last quarter.

3.3 Sector-Relative Target

Initial experiments using a raw target (*"Did stock beat S&P 500?"*) yielded poor precision (∼40%) because the target was dominated by market beta (possibly the overall market direction into bullish tech stocks).

I then pivoted to a Sector-Relative Target. The target variable y is defined as:

$$y_{i,t} = \begin{cases} 1 & \text{if } Return_{i,t+63} > \text{Median}(Return_{Sector,t+63}) \\ 0 & \text{else} \end{cases}$$

This "curves the grades," forcing the model to find the best Utility stock among Utilities or the best Tech stock among Tech. This guarantees a balanced dataset (50% winners, 50% losers) regardless of market conditions.

3.4 Model Architecture

I selected the **Histogram Gradient Boosting Classifier** (`HistGradientBoostingClassifier` from scikit-learn).

- **Why Gradient Boosting?** Unlike Random Forests, which build trees independently, Gradient Boosting builds trees sequentially to correct the errors of previous trees. This makes it superior for detecting subtle signals in noisy financial data.
- **Training Regime:**
  - **Training:** 2015–2023 (Learning historical patterns).
  - **Testing:** 2024–2025 (Out-of-sample validation).
  - **Optimization:** `RandomizedSearchCV` was usde to find the best hyperparameters. This method randomly samples parameters from a grid, which is faster than checking every combination.The given variables were tuned:

∗ `learning_rate`: Controls the contribution of each tree. A lower rate reduces the risk of overfitting but requires more trees.

∗ `max_depth`: Limits the maximum depth of the individual trees. This controls the complexity of the model; deeper trees can model more complex patterns but risk memorizing noise.

∗ `min_samples_leaf`: Sets the minimum number of samples required to be at a leaf node. Higher values smooth the model, preventing it from isolating outliers.

3.5 Strategic Allocation Profiles

The dashboard allows users to select distinct risk profiles (Growth, Balanced, Income). These profiles define the base asset allocation before the machine learning model applies its active weights.

The portfolio is constructed using a robust set of underlying assets to ensure diversification:

- **Equities (Active):** A universe of ∼320 stocks across all 11 GICS sectors.
- **Fixed Income (Passive):** A diversified bond mix using **AGG** (US Core Bonds), **LQD** (Investment Grade Corporate), and **HYG** (High Yield).
- **Alternatives (Passive):** Inflation hedges including **GLD** (Gold), **DBC** (Commodities Index), and **VNQ** (Real Estate REITs).
- **Liquidity: SHV** (Short-Term US Treasury) for risk-free stability.

Profile	**Equities**	**Fixed Income**	**Liquidity**	**Alternatives**
Growth	70%	20%	5%	5%
Balanced	50%	40%	5%	5%
Income	30%	60%	5%	5%

Table 1: Strategic asset allocation mixes.

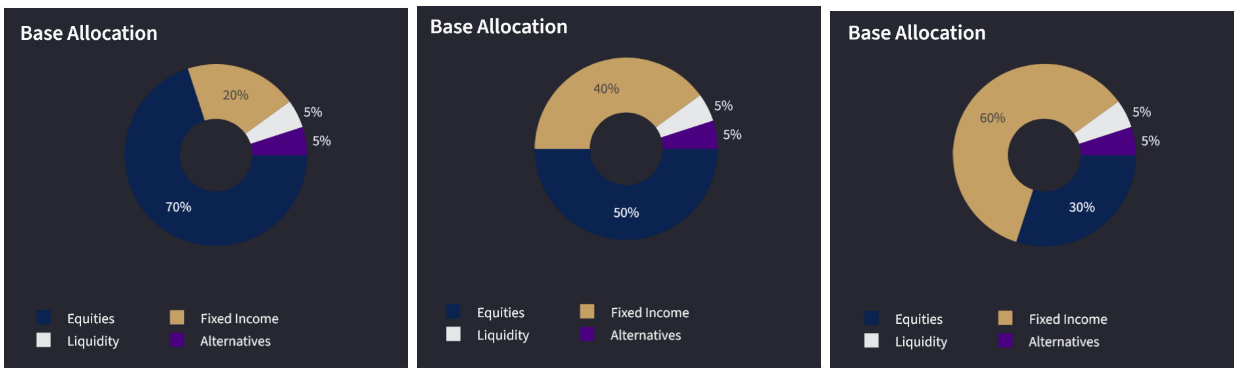


Figure 1: Strategic asset allocation visual breakdown for Growth, Balanced, and Income profiles.

3.6 Portfolio Construction & Monte Carlo

The model outputs a probability score (Confidence). Instead of buying all stocks with $> 50\%$ probability, I implement a **Top Decile Strategy**:

1. **Filter:** Only consider stocks in the top 10–20% of confidence scores.
2. **Aggressive Weighting:**
   - **High Conviction (Signal 1):** Weight multiplier = **5.0x** (Overweight).

- **Low Conviction (Signal 0):** Weight multiplier = **0.0x** (Sell/Avoid).

3. **Monte Carlo Simulation:** To project future performance, the system executes a Monte Carlo simulation generating 200 independent price paths over a 252-day trading horizon. The simulation relies on `numpy.random.multivariate_normal` to generate returns based on the historical covariance matrix of the selected assets. This methodology accounts for the correlation between assets (e.g., if Tech stocks crash, they likely move together), ensuring the risk projections are mathematically consistent with historical inter-asset relationships.

4 Implementation

The project is structured as a modular Python pipeline in the `src/` directory.

4.1 Key Technical Decisions

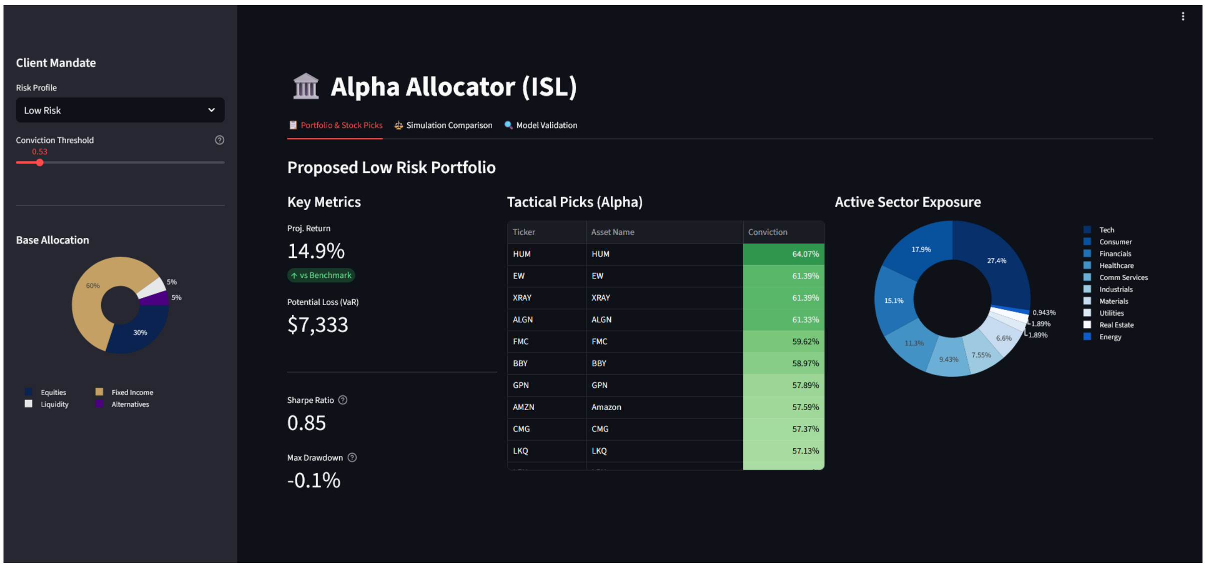
- **Dynamic Thresholding:** In `optimize_model.py`, the model scans the validation set for the probability threshold that maximizes precision.
- **Survivorship Bias Mitigation:** In `portfolio_simulation.py`, the code dynamically cleans the asset universe. If a stock (e.g., UBER) did not exist in 2016, the covariance matrix calculation automatically excludes it or trims the simulation window to valid data points to prevent mathematical convergence errors.
- **Data Leakage Prevention:** Features are calculated using *lagged* data (available at time t), while targets look forward to $t + 63$. The train/test split is strictly temporal (no shuffling).

4.2 Software Stack

Pandas/NumPy for data manipulation, **Scikit-Learn** for Gradient Boosting, **Plotly** for interactive charts, and **Streamlit** for the web dashboard.

4.3 Dashboard Interface

The Streamlit dashboard serves as the control center. It allows users to visualize portfolio composition, run simulations, and audit model accuracy.



Client Mandate

Risk Profile: Low Risk

Conviction Threshold: 0.53

Proposed Low Risk Portfolio

Key Metrics

Proj. Return: 14.9% (↑ vs Benchmark)

Potential Loss (VaR): $7,333

Sharpe Ratio: 0.85

Max Drawdown: -0.1%

Tactical Picks (Alpha)

Ticker	Asset Name	Conviction
HUM	HUM	64.07%
EW	EW	61.39%
XRAY	XRAY	61.39%
ALGN	ALGN	61.33%
FMC	FMC	59.62%
BBY	BBY	58.97%
GPN	GPN	57.89%
AMZN	Amazon	57.59%
CMG	CMG	57.37%
LKQ	LKQ	57.13%

Figure 2: The Alpha Allocator Dashboard Interface.

Key Feature: Conviction Threshold Slider

The "Conviction Threshold" slider controls the risk tolerance of the AI model.

- **Higher Threshold (e.g., 0.60):** The model forces extremely selective trading (high confidence only), increasing precision but reducing diversification.
- **Lower Threshold (e.g., 0.51):** The model trades more broadly, increasing diversification but potentially lowering the statistical edge per trade.

5 Codebase & Reproducibility

5.1 How to Run

1. **Environment:** The project includes a `requirements.txt` file listing dependencies.
2. **Execution:** Run `python main.py`. This script automatically checks dependencies, runs data processing, trains the model, and launches the dashboard.

5.2 Reproducibility Features

- **Random State:** Fixed to `42` in all ML models and simulations.
- **Asset Handling:** Robust error handling for `yfinance` downloads prevents crashes if a ticker is delisted.

6 Results

6.1 Model Performance

The Gradient Boosting model, utilizing the Sector-Relative target, achieved impressive results on the 2024–2025 out-of-sample test set.

- **Global Accuracy:** ∼52%
- **Precision (Win Rate): 54.6%** on the Top 10% high-confidence picks.

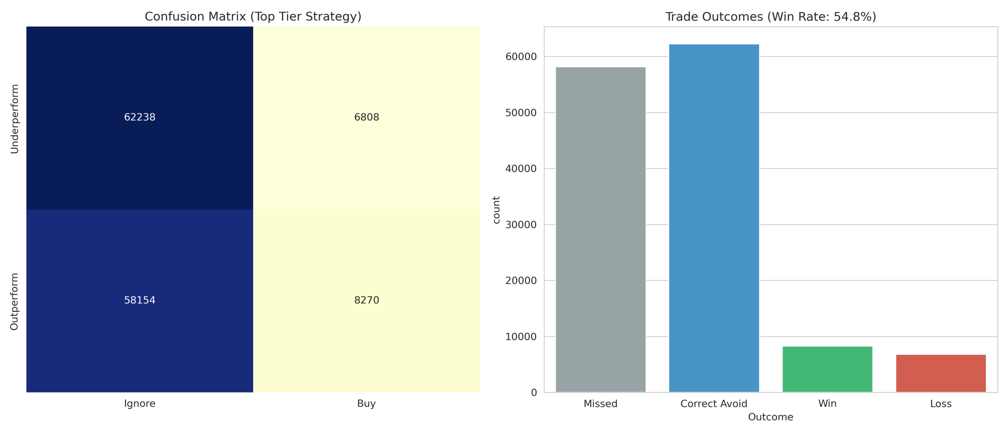


Figure 3: Model Audit showing Confusion Matrix and Outcome Distribution.

6.2 Simulation Results (Alpha Generation)

The Monte Carlo simulation (Figure 7) demonstrates the impact of this precision for a **Balanced Risk Profile**. By concentrating capital into the top decile of stocks and aggressively cutting losers, the Monte Carlo simulation projects a higher expected value for the Active Portfolio compared to the Benchmark.This suggests that the model's stock selection concentrates exposure in assets with higher historical drift, though this projection assumes historical correlations persist.

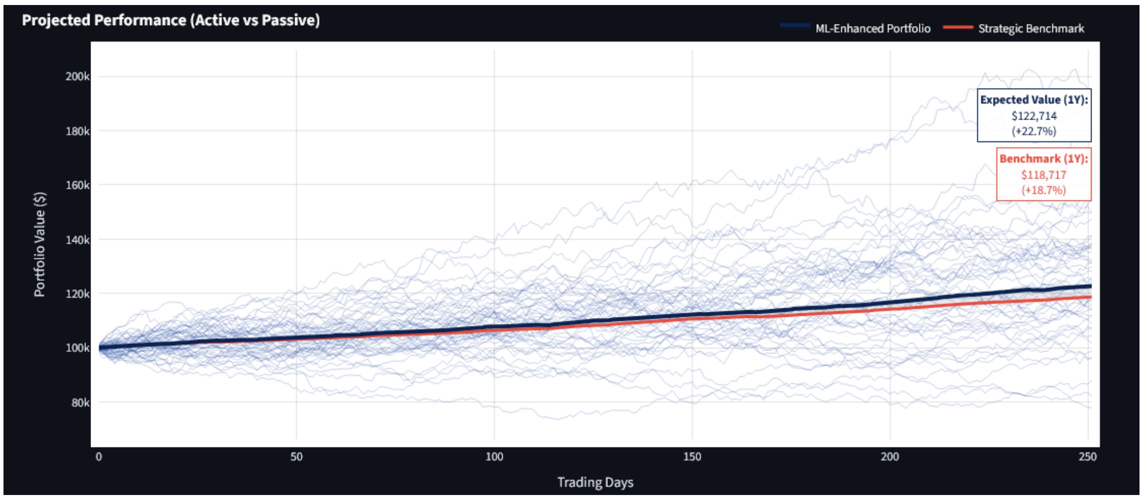


Figure 4: Portfolio Simulation (Balanced Risk Profile)

7 Conclusion

7.1 Summary

The allocator successfully demonstrates that Machine Learning can extract alpha from public market data if the problem is framed correctly. The shift from "Absolute Prediction" to "Sector-Relative Ranking" was the turning point, allowing the model to identify quality assets regardless of broader market conditions.

7.2 Limitations

- **Transaction Costs:** The simulation assumes free trading. High turnover could erode alpha.
- **Market Regime:** The test set (2024–2025) was largely a bull market. Performance in a severe recession remains to be fully stress-tested.
- **Survivorship Bias:** The study utilizes a static asset universe based on current S&P 500 constituents. This introduces survivorship bias, as companies delisted during the training period are excluded. Future iterations should utilize point-in-time constituent lists to mitigate this." Acknowledging it turns a "mistake" into a "known constraint.

7.3 Future Work

- **Sentiment Analysis:** Integrating NLP scores from financial news.
- **Risk Parity:** Implementing sophisticated weighting schemes beyond simple multipliers.

References

[1] Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1), 3–56.

[2] Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5), 2223–2273.

[3] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[4] Levy, R. A. (1967). Relative Strength as a Criterion for Investment Selection. *The Journal of Finance*, 22(4), 595–610.

Appendix: AI Tools Usage

This project utilized AI assistance (Gemini) for: 1. **Debugging:** Resolving `numpy.linalg.LinAlgError` (SVD convergence) in the simulation. 2. **Refactoring:** Optimizing the `data_processing.py` loop. 3. **Visualization:** Generating Plotly code for the simulation chart. 4. **Research Data**: Searching relevant research that could be employed into the ML model.