Machine Learning-Optimized Pairs Trading Strategy

QF621 Sample Report - Group 8

Executive summary

This project advances pairs trading strategy by integrating machine learning (ML) to overcome limitations faced when deploying it in the traditional way. Pairs trading strategy seeks to profit from temporary price divergences between assets that are cointegrated by buying the asset that is relatively cheaper and simultaneously selling the more expensive one.

However, challenges such as computational inefficiencies in searching the entire stock universe for cointegrated pairs have hindered its potential, especially for smaller firms without sophisticated resources and technology tools. The search process takes very long, rendering the strategy less deployable in real trading.

This project incorporated Bayesian Optimization (BO) to identify cointegrated pairs and then used the XGBoost algorithm to forecast the price spread between the identified cointegrated pairs.

Bayesian Optimization:

A scoring objective function optimizes BO by incorporating SMA correlation and volatility, emphasizing mean reversion and trend alignment. Adjustable alpha and beta values (0.2 and 0.1) ensured stationarity in selected pairs while balancing trend and volatility impacts.

A greedy search loop combined with BO identifies statistically significant pairs efficiently by minimizing cointegration p-values, addressing resource constraints compared to a brute force pairwise search.

XGBoost:

After some feature engineering, we leveraged on the XGBoost algorithm to forecast next-day spread changes which will then be used to generate the trading signals.

Overall, by combining Bayesian Optimization and XGBoost, this ML-optimized pairs trading strategy presents a scalable and profitable approach, addressing computational challenges and providing robust diversification by trading multiple pairs at the same time. It represents a significant advancement in creating actionable and efficient trading opportunities as it streamlined the end-to-end process and generated superior profits.

Problem statement

The pairs trading strategy was first developed in the 1980s by Morgan Stanley's quantitative team. This popular market-neutral strategy leverages on profiting off temporary price divergence between 2 historically cointegrated assets believing that they will eventually converge to their long-run equilibrium relationship (Hudson and Thames, n.d.).

However, as this strategy became more widespread, profit margins from this strategy eroded away. Furthermore, the success of this strategy is dependent on identifying cointegrated stocks. The classical approach of doing a pairwise search on a stock universe is inefficient as it will iterate through a total of NC2 pair; with a complexity of O(n^2) which further increases the computational exhaustion in GPUs which is largely detrimental for small size trading firms who do not have enhanced infrastructure and servers. Moreover, it also involves choosing the right hyperparameters like length of training period, and z-score thresholds for entry and exit etc

Thus, the objective of this project is to develop a pairs trading strategy using machine learning (ML) techniques such as Bayesian Optimisation and XGBoost. Bayesian Optimisation is done to select and identify cointegrating pairs of stocks while XGBoost is deployed to predict the next day's spread. The ultimate aim is to efficiently identify pairs of stocks and boost profits or reduce downside risk.

Overview of Pair Trading Strategy with Machine Learning

Close Prices are used instead of Returns

To begin our classical approach for identifying cointegrated pairs, we extracted daily close prices for all S&P 500 constituents from 2019 to 2023 as part of training data as well as 2024 to test on the Machine Learning Model. In the context of cointegration, we specifically use close prices rather than returns. This is because returns are typically stationary by nature and exhibit limited long-term drift, whereas close prices are generally non-stationary and integrated of order 1 (Falk & Lens, 2005). Since cointegration analysis is designed to uncover long-run equilibrium relationships between non-stationary time series, applying it to returns would be inappropriate and could yield misleading or inconclusive results.

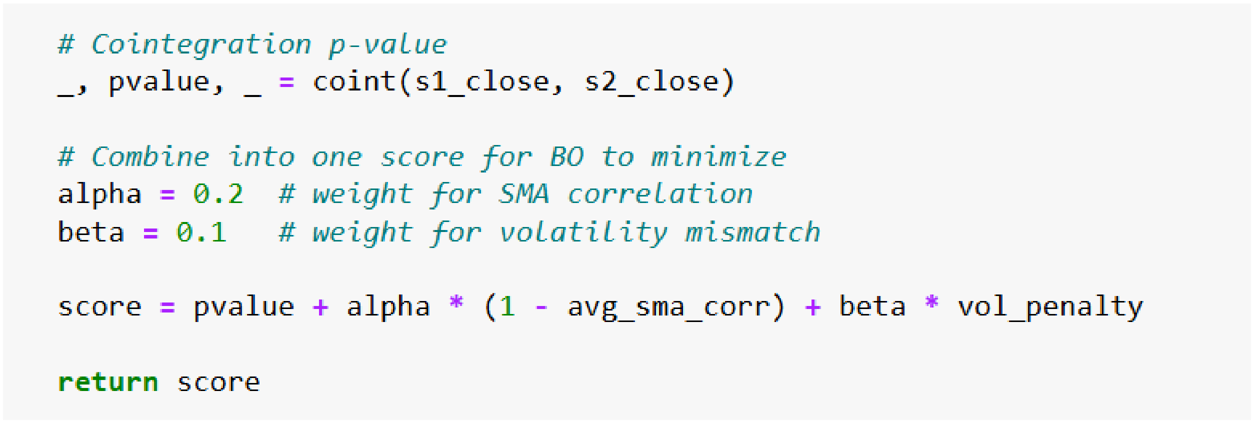
After obtaining the data , we proceeded to create the following user defined functions that are essential for us to ultimately extract the cointegrated pairs for the year.

a. Prepare_close_prices_and_pairs : Cleaning of the Price Data
b. Calculate_indicators : Returning a DataFrame containing Close Prices, SMA-5, SMA-10, and Volatility.
c. Find_cointegrated_pairs: Returns a list of tuples containing cointegrated pairs and their scores after running Bayesian Optimisation

Defining a Scoring Objective Function to identify Cointegrated Pair

In this case, the sub-function “cointegration_test” (under “find_cointegrated_pairs”) generates a score for the identified pair. Since we are targeting stationarity, a lower score for the identified pair will mean a higher stationarity between the pair.

In addition, to optimize the accuracy of the objective function , we included SMA and Volatility correlation as input parameters in order for the BO to learn and obtain the best minimized p-value. These 2 indicators are used due to our main focus on this strategy which is on mean reversion with a huge relation to trend following using SMA as well as Volatility which can give us deeper insights to the z-score towards our trade frequency.



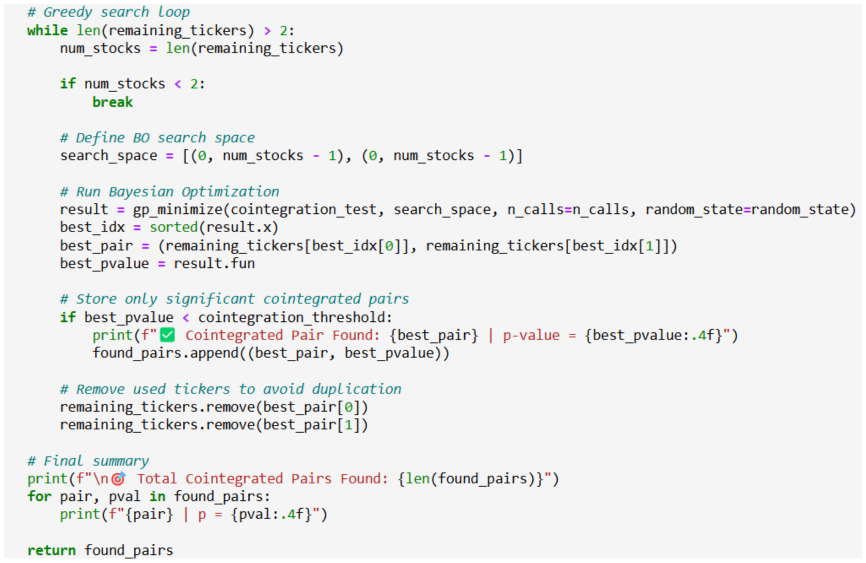
```
# Cointegration p-value
_, pvalue, _ = coint(s1_close, s2_close)

# Combine into one score for BO to minimize
alpha = 0.2  # weight for SMA correlation
beta = 0.1   # weight for volatility mismatch

score = pvalue + alpha * (1 - avg_sma_corr) + beta * vol_penalty

return score
```

Figure 1: Identify cointegrating pairs

Alpha and beta are assigned to the SMA correlation and volatility respectively, allowing us to control the relative importance of each indicator in our scoring function for Bayesian Optimization. We set the default values to alpha = 0.2 and beta = 0.1, reflecting a greater emphasis on short- to mid-term trend alignment over volatility similarity. This is aligned with our strategy’s focus on identifying mean-reverting spreads. Ultimately, a lower score from the function indicates a more stationary pair, as it reflects a lower penalty on pairs with stronger SMA correlations and tolerable levels of volatility divergence.

Bayesian Optimisation

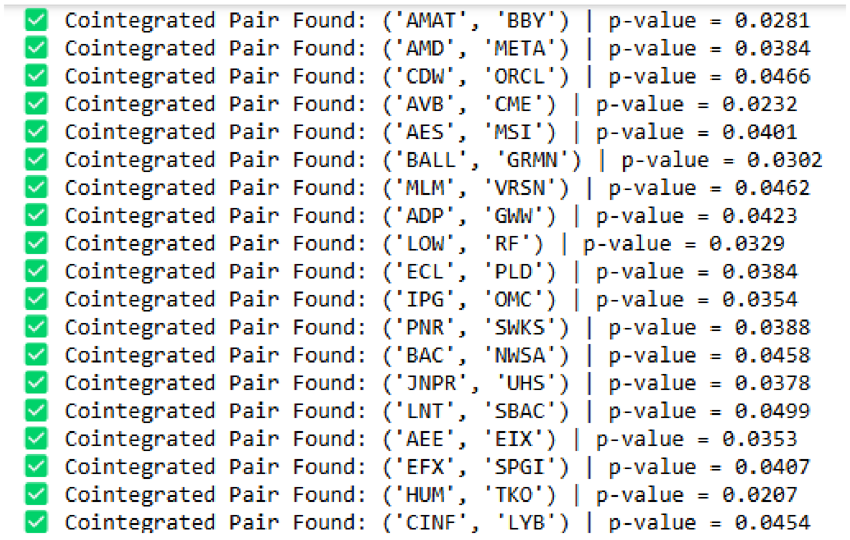
To identify cointegrated pairs among a list of ticker symbols, the code implements a greedy search loop combined with Bayesian Optimization (BO). The loop continues to run as long as there are at least two tickers left in the remaining_tickers list. In each iteration, BO is used to efficiently search for the best pair of stocks by minimizing the p-value from a cointegration test, using a defined search space over the index positions of the tickers. If the best p-value returned is below a specified significance threshold, the corresponding stock pair is considered statistically cointegrated and is stored in the found_pairs list. In addition, to diversify our portfolio and prevent excessive weight on particular tickers, both tickers are removed from the pool, reducing the search space for the next iteration. This process continues until no more valid pairs can be formed.



```python
# Greedy search loop
while len(remaining_tickers) > 2:
    num_stocks = len(remaining_tickers)

    if num_stocks < 2:
        break

    # Define BO search space
    search_space = [(0, num_stocks - 1), (0, num_stocks - 1)]

    # Run Bayesian Optimization
    result = gp_minimize(cointegration_test, search_space, n_calls=n_calls, random_state=random_state)
    best_idx = sorted(result.x)
    best_pair = (remaining_tickers[best_idx[0]], remaining_tickers[best_idx[1]])
    best_pvalue = result.fun

    # Store only significant cointegrated pairs
    if best_pvalue < cointegration_threshold:
        print(f"✅ Cointegrated Pair Found: {best_pair} | p-value = {best_pvalue:.4f}")
        found_pairs.append((best_pair, best_pvalue))

    # Remove used tickers to avoid duplication
    remaining_tickers.remove(best_pair[0])
    remaining_tickers.remove(best_pair[1])

# Final summary
print(f"\n🎯 Total Cointegrated Pairs Found: {len(found_pairs)}")
for pair, pval in found_pairs:
    print(f"{pair} | p = {pval:.4f}")

return found_pairs
```

Figure 2: Deploying Bayesian Optimisation

At the end, the code outputs a summary of all significant cointegrated pairs identified, making the approach both statistically rigorous and computationally efficient. The key takeaway from this phase is that Bayesian Optimization significantly accelerates the pair selection process, offering a practical alternative to brute-force testing. Although it may sacrifice some degree of accuracy compared to the exhaustive brute force pairwise search method, it effectively addresses the constraints faced by smaller firms and retail investors who lack the infrastructure and computing power to perform full-scale cointegration analysis.



```
✅ Cointegrated Pair Found: ('AMAT', 'BBY') | p-value = 0.0281
✅ Cointegrated Pair Found: ('AMD', 'META') | p-value = 0.0384
✅ Cointegrated Pair Found: ('CDW', 'ORCL') | p-value = 0.0466
✅ Cointegrated Pair Found: ('AVB', 'CME') | p-value = 0.0232
✅ Cointegrated Pair Found: ('AES', 'MSI') | p-value = 0.0401
✅ Cointegrated Pair Found: ('BALL', 'GRMN') | p-value = 0.0302
✅ Cointegrated Pair Found: ('MLM', 'VRSN') | p-value = 0.0462
✅ Cointegrated Pair Found: ('ADP', 'GWW') | p-value = 0.0423
✅ Cointegrated Pair Found: ('LOW', 'RF') | p-value = 0.0329
✅ Cointegrated Pair Found: ('ECL', 'PLD') | p-value = 0.0384
✅ Cointegrated Pair Found: ('IPG', 'OMC') | p-value = 0.0354
✅ Cointegrated Pair Found: ('PNR', 'SWKS') | p-value = 0.0388
✅ Cointegrated Pair Found: ('BAC', 'NWSA') | p-value = 0.0458
✅ Cointegrated Pair Found: ('JNPR', 'UHS') | p-value = 0.0378
✅ Cointegrated Pair Found: ('LNT', 'SBAC') | p-value = 0.0499
✅ Cointegrated Pair Found: ('AEE', 'EIX') | p-value = 0.0353
✅ Cointegrated Pair Found: ('EFX', 'SPGI') | p-value = 0.0407
✅ Cointegrated Pair Found: ('HUM', 'TKO') | p-value = 0.0207
✅ Cointegrated Pair Found: ('CINF', 'LYB') | p-value = 0.0454
```

Figure 3: Identified Cointegrated Pairs

Overview of ML-Optimised Pairs Trading strategy

Introduction of our Forecasting Trading Strategy

The ML-optimised pairs trading strategy aims to identify profitable trading opportunities by forecasting the next day's spread between cointegrated asset pairs using the gradient boosting tree model XGBoost. XGBoost is a widely used model for both regression and classification tasks. Its feasibility is well supported by our current literature review, with studies showing how the use of XGBoost generated returns of up to 23% p.a. reported by Figueira & Horta (2022), 60% in-sample accuracy in classifying profitable trades shown by Jirapongpan & Phumchusri (2020), and superior performance over deep neural networks and market benchmarks, achieving 46% p.a. returns (Krauss et al., 2016).

At the core of the strategy is the prediction of the spread change, denoted as $\Delta_{t+1} = \hat{s}_{t+1} - s_t$, where s represents the spread at time t. This formulation enables the model to anticipate whether the spread will widen or narrow, allowing us to make trading decisions. When the difference in forecasted and current spread suggests a significant deviation from the current level (accounting for transaction costs), the strategy takes a long or short position accordingly. Our project draws inspiration from the works of Figueira and Horta (2022) and Hadad et al. (2024), the approach leverages a regression model to forecast the next day spread ratios and utilise it in our trading rules (Figure 4). The only difference in our trading rule is defined as follows.

1) We remain in a long/short position if the predicted signal is equivalent to the previous day prediction.
2) If the signal strength is not strong enough (spread difference lower than the TC), we will exit the position if we were in a current position.

$$\hat{P} = \begin{cases} \text{if } \Delta_{t+1} > \text{TC}, \text{ Long position,} \\ \text{if } \Delta_{t+1} < -\text{TC}, \text{ Short position,} \\ \text{otherwise, remain outside the market.} \end{cases}$$

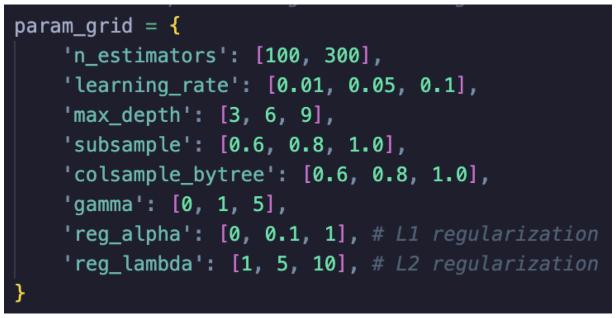
Figure 4: Formulation of Trading Rule (Figueira & Horta, 2022)

Hyperparameter tuning of the XGBoost Regressor

We extracted yearly cointegrated pairs from 2019 to 2023 and generated features for each using a 30-day rolling window. These features included statistical relationships , spread-based metrics, mean reversion indicators, technical signals, dynamic risk measures and lead-lag differences in returns (Table 1). Using the features we engineered, we trained our XGBoost Model and determined the optimal model hyperparameters using a validation set (2023 co-integrated pairs and 2023 spreads data). In total, we ran across 4374 permutations (Figure 5).

Rolling Window: 30 Days		
Category	**Feature Name**	**Description**
Statistical Relationship	hedge_ratio	Static beta from OLS regression of Price A on Price B
	alpha	Intercept term from OLS regression
Spread-Based Features	spread	Price A − β × Price B − α
	spread_z	Z-score of the spread
	spread_vol	Rolling standard deviation of spread
	next_day_spread	Spread value for the next day (TARGET VARIABLE)
Mean Reversion Metrics	halflife	Time for spread to revert halfway to its mean
Technical Indicators	bollinger_pct	Spread's position within ±2 std Bollinger Bands
Dynamic Risk Measures	rolling_beta	Time-varying beta over a rolling window
	vol_ratio	Ratio of Price A vs. Price B volatilities
Lead-Lag Relationship	lead_lag_diff	Lagged difference in percentage returns

Table 1: Feature Engineering Table



```
param_grid = {
    'n_estimators': [100, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 6, 9],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'gamma': [0, 1, 5],
    'reg_alpha': [0, 0.1, 1], # L1 regularization
    'reg_lambda': [1, 5, 10], # L2 regularization
}
```

Figure 5: Hyperparameter Tuning Parameter Grid for XGBoost Regressor

Evaluating our Forecasting Model

To prevent look-ahead bias, we evaluated our forecasting model and trading strategy on 2024 price test set data using the set of cointegrated pairs identified in 2023. This assumes that the relationships observed in 2023 persist into 2024. By comparing the predicted spreads against the actual spreads (ground truth), we were able to visually assess the model's ability to anticipate next-day spread movements and generate timely trading signals. This is not surprising, considering we used the current day spread features as a feature in our forecasting model.

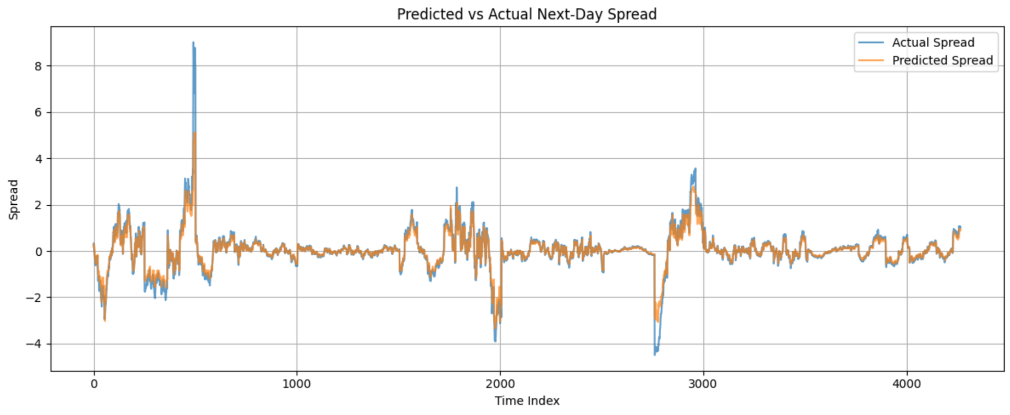


Figure 6: Predicted vs Actual Spreads on Test Set

Results of the ML Pair Trading Strategy

To assess the performance of our trading strategy, we make use of metrics such as Max Drawdown (MDD), Sharpe ratio, and returns. CAGR is not needed here since our test period is exactly 1 year long.

These metrics give a holistic assessment of returns and risk. We will benchmark the ML-optimised strategy against the classical approach and buy-and-hold in the following segment of the report.

As mentioned, we ran our trading strategy on the 2023 Co-integrated pairs and analysed its performance on its 2024 Price Data. Across all the 17 2023-Co-integrated pairs, we generated a return of 19.81% at the aggregate level, with a Sharpe ratio of 2.53 and MDD of 3%.

Figure 7 below shows our out of sample performance as plotted on a wealth curve.

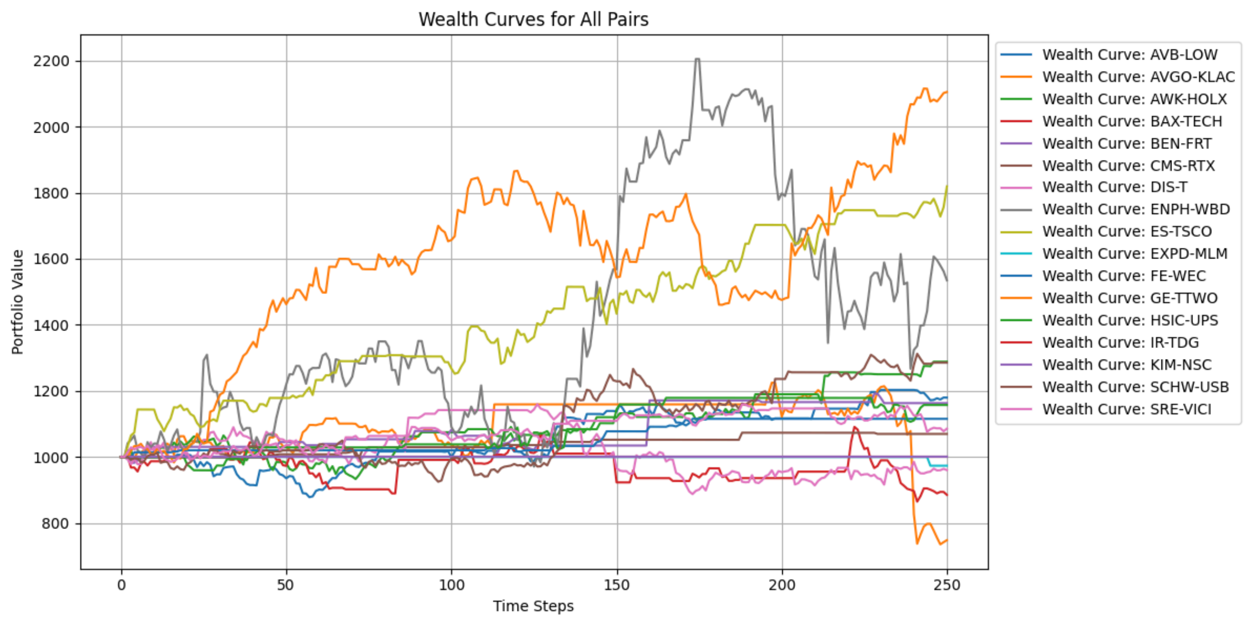
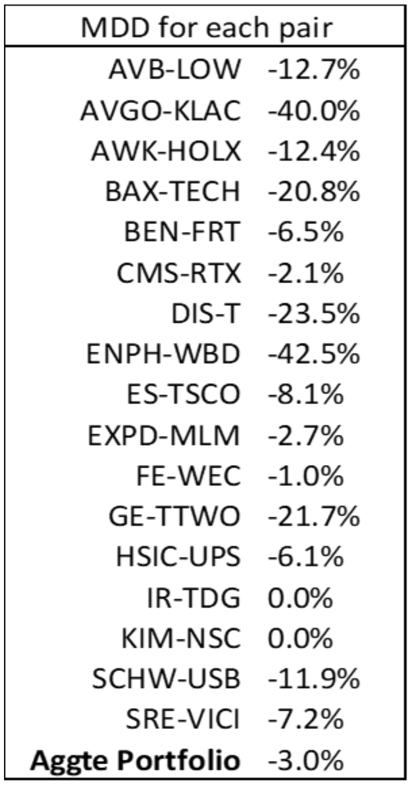


Figure 7: Wealth curve for each individual pairs

This wealth curve plots the cumulative profit and loss for each pair over the test period of 2024, reflecting how each pair performed during this period with an initial capital of $1,000 for each pair. Some pairs had a wealth curve that flat lined at $1,000 since the start of the testing period as our strategy generated no signal for them during the test period at all. Overall, we see largely positive returns at the portfolio aggregate level.

Portfolio risk

Holding a portfolio of 17 pairs significantly reduced our downside risk as proxied by MDD. Since each pair is uncorrelated with one another, they did not drawdown simultaneously. Figure 8 below shows MDD of our strategy on the 2024 test set:



MDD for each pair	
AVB-LOW	-12.7%
AVGO-KLAC	-40.0%
AWK-HOLX	-12.4%
BAX-TECH	-20.8%
BEN-FRT	-6.5%
CMS-RTX	-2.1%
DIS-T	-23.5%
ENPH-WBD	-42.5%
ES-TSCO	-8.1%
EXPD-MLM	-2.7%
FE-WEC	-1.0%
GE-TTWO	-21.7%
HSIC-UPS	-6.1%
IR-TDG	0.0%
KIM-NSC	0.0%
SCHW-USB	-11.9%
SRE-VICI	-7.2%
Aggte Portfolio	-3.0%

Figure 8: Risk Metrics (Individual pairs vs aggregate level)

Though some pairs exhibited deep drawdowns with MDD reaching 40% and 42.5%, our portfolio only had a MDD of 3%. The above example showed how diversification managed to shield our portfolio against extreme losses that come from specific pairs. Our selection methodology proved to be effective at functioning as a natural safeguard against excessive risk exposure.

Figure 9 has been plotted to show the wealth curve of the portfolio at an aggregate level. Overall, we can see that wealth grew steadily with a rather muted drawdown.

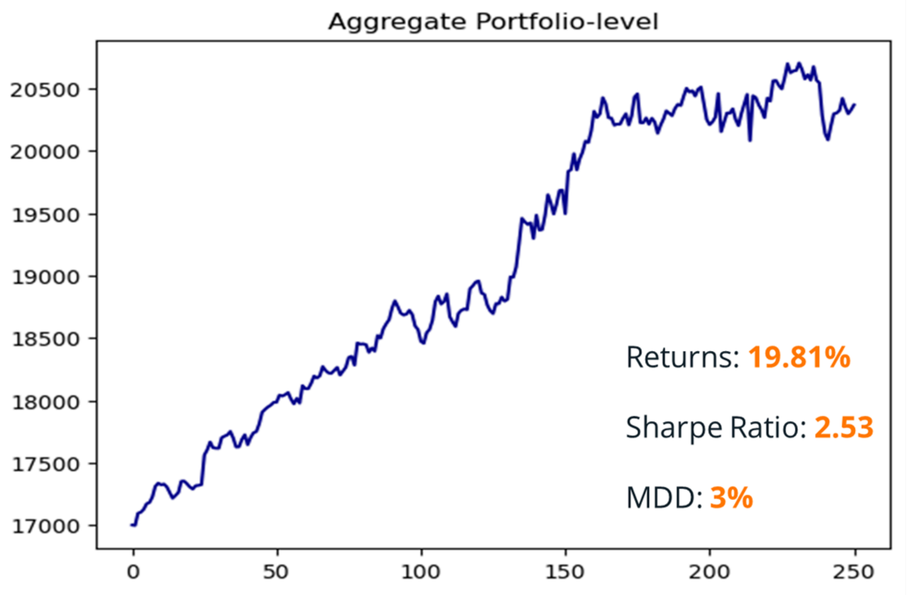


Figure 9: Aggregate portfolio wealth curve

Results of the Trading Strategy on selected pairs
To analyse our strategy's performance further, we selected a few pairs as shown below. We added glyphs to the plot to visualise the effect of trading decisions on the wealth curve.

Figure 10 below shows the test performance of the pair "SRE-VICI". Our strategy generated profits steadily despite periods where the spreads did not exhibit stationarity in the test period. It also experienced deep drawdown between day 150 to 200, where we long the spread expecting it to converge but it diverged further.

Figure 11 below shows the test performance of the pair "AWK-HOLX". Our strategy suffered a huge drawdown between day 25 to 70 as the cointegrating relationship between the pair broke down, as shown by the downward trending spread. However, when the long-run equilibrium relationship between the pairs regained, our strategy performed consistently well with little drawdown risk.

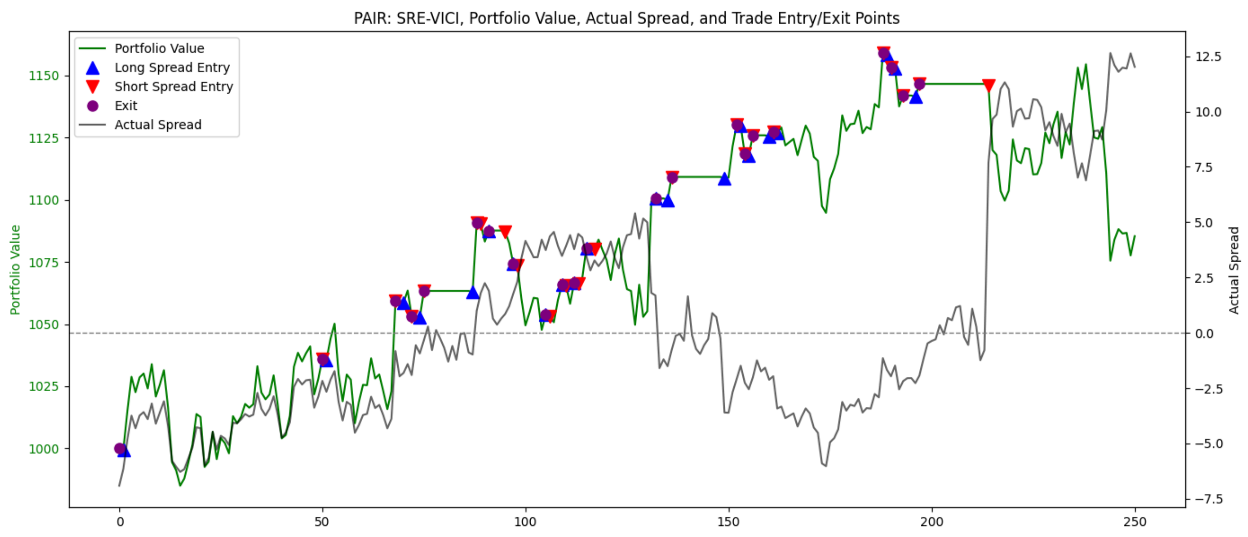


Figure 10: Pair: SRE-VICI test performance

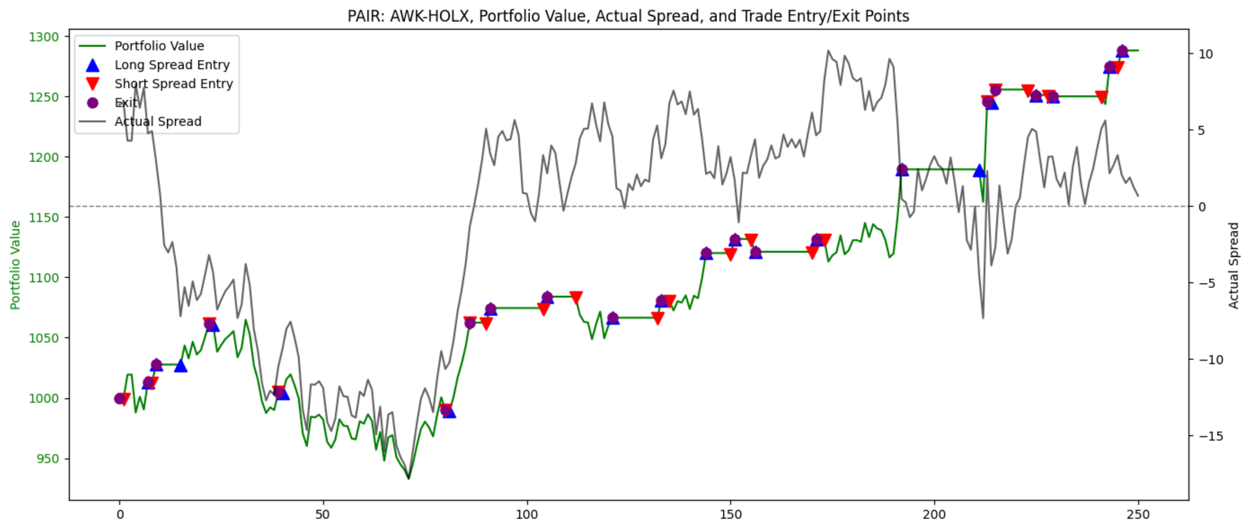


Figure 11: Pair: AWK-HOLX test performance

Baseline Trading Strategies

Now that we've discussed how the machine learning-based pairs trading strategy works, we would like to evaluate how it performs in comparison to several baseline strategies. This comparison helps us understand whether the added complexity of machine learning actually leads to better trading outcomes, or if simpler strategies might be just as effective.

One of the strategies we use as a benchmark is the buy-and-hold strategy. This is a long-term investment approach where an investor buys a financial asset and holds onto it for an extended period, regardless of short-term market fluctuations. The idea behind this strategy is that, over time, markets generally trend upwards, even though there may be periods of volatility along the way.

This strategy is often used as a baseline because research has shown that many active trading strategies fail to consistently outperform a simple buy-and-hold approach, especially after accounting for fees and taxes. So if a new strategy cannot beat buy and hold in terms of long-term returns or risk-adjusted performance, it is often considered not worth pursuing.

Figure 12 below shows the performance of a buy-and-hold strategy for the S&P 500 for 2024. During this period, the portfolio increased in value from approximately \$100,000 to about \$119,425, representing a total return of 19.4%.

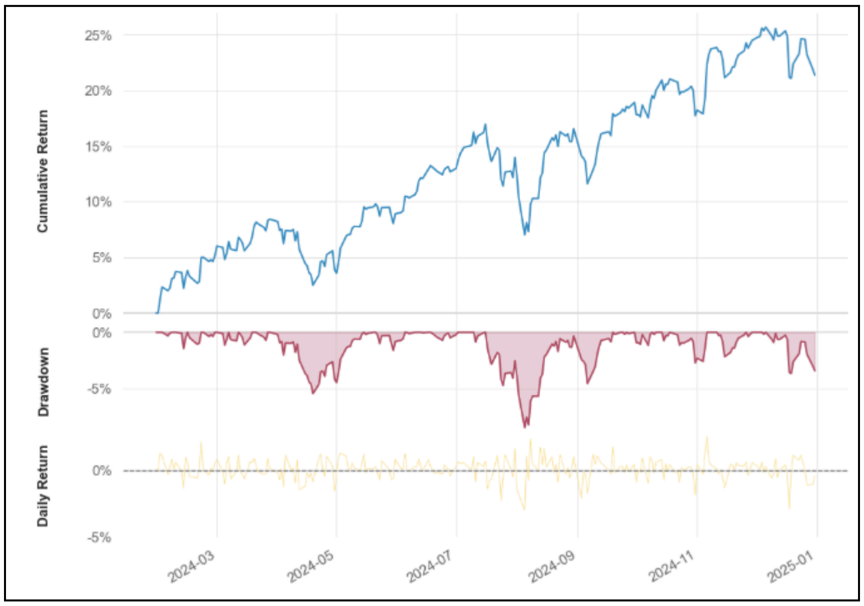


Figure 12: Pair: Cumulative Returns, Drawdown and Daily Returns of S&P 500 for 2024

While there were some short-term declines most notably in April and again in late August, the overall trend was upward. The maximum drawdown during this time was -8.5%.

Finally, the Sharpe ratio was 1.71. This indicates strong risk-adjusted returns, meaning the strategy delivered relatively high returns for the amount of risk taken.

Another baseline strategy that we will use for comparison with the machine learning-based approach is the trend-following strategy. This approach tries to catch upward or downward trends using indicators like moving averages. A common example is the moving average crossover in which we buy when the shorter moving average goes above the longer moving average and sell when it drops below.

Unlike buy and hold, trend-following aims to reduce risk by exiting or reversing positions during downturns. Because of that, it's often used as a benchmark for more defensive trading strategies.

Figure 13 below shows the S&P 500 with its 5-day and 20-day moving averages. We've marked the points where the two lines cross, which are used as buy or sell signals in the trend-following strategy.

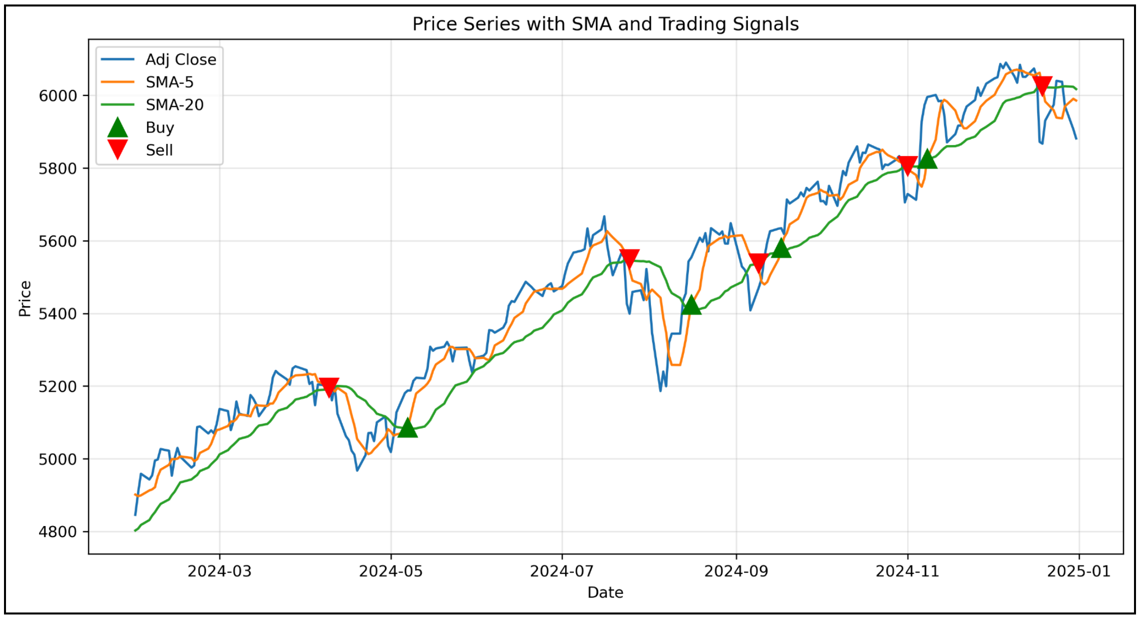


Figure 13: S&P 500 Price Alongside 5-Day and 20-Day Moving Averages with Buy and Sell Signals

Figure 14 below shows how the trend-following strategy performed in 2024.

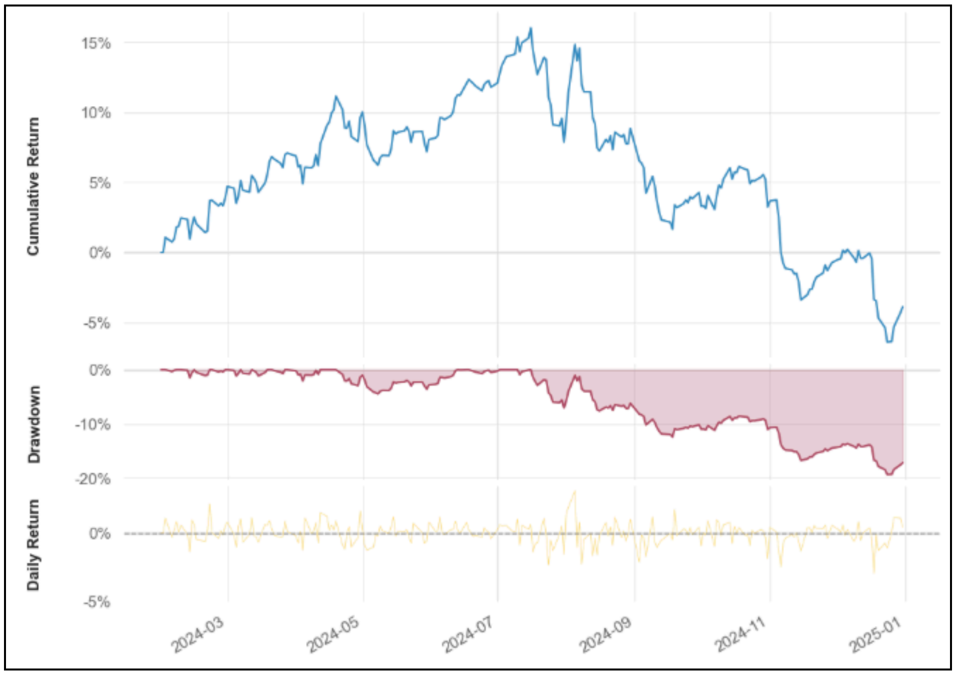


Figure 14: Performance of Trend Following Strategy for 2024

Looking at the cumulative return, we see that the strategy did quite well in the first half of the year. By July, it reached a peak return of around 16%. But from August onwards, performance dropped sharply. By the end of the year, most of the gains were lost.

Comparison of Strategies

Now that we've gone through our baseline strategies, let's compare their performance to see how they stack up against each other.

Figure 15 below shows a comparison of the investment strategies we have just mentioned over the course of 2024, assessing their performance in terms of their returns, Sharpe ratio, and maximum drawdown.

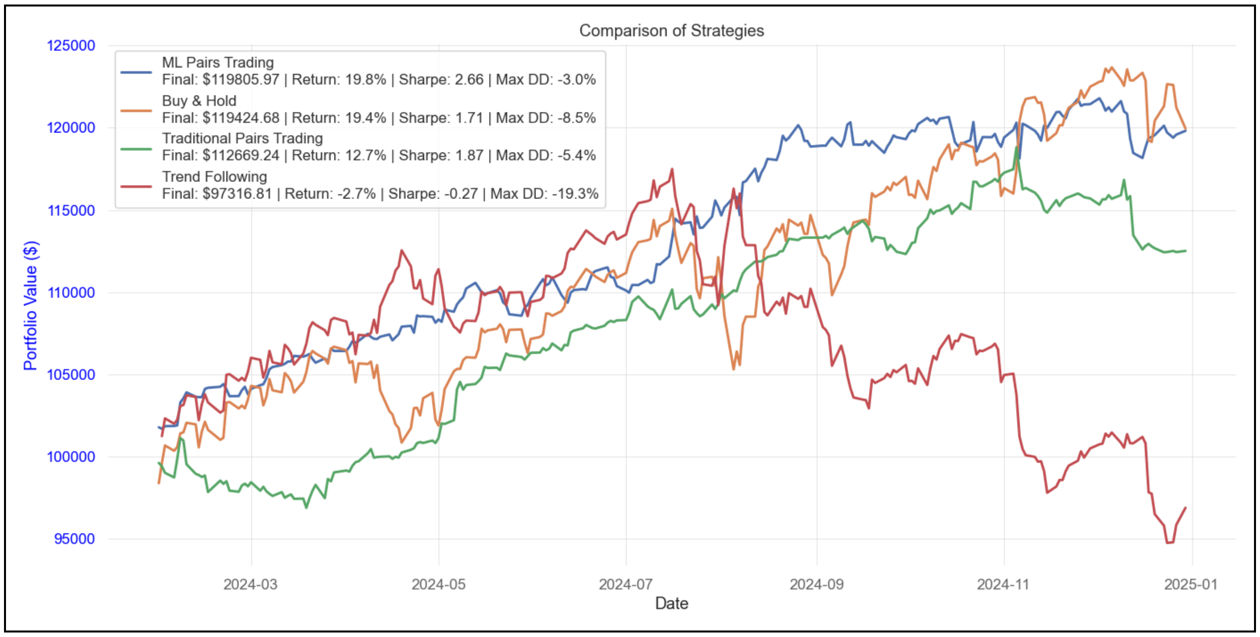


Figure 15: Comparison of Strategies

As we can see from the graph, the ML-Based Pair Trading Strategy is the top performing strategy since it offers nearly the same return as Buy & Hold but with lower volatility and less risk.

This suggests that the machine learning strategy not only achieves strong returns, but also manages risk more effectively than the other approaches.

Limitations of ML Based Strategy

Despite its strong performance, the machine learning-based strategy does come with some limitations.

Firstly, in this project, we used daily data to build and test the model. However, in practice, this type of strategy is often used in high-frequency trading, where it performs better on intraday time frames, such as hourly or even minute-level data. This means that with access to higher-frequency data, the strategy could potentially be even more effective

The second limitation to note is the effect of regime shifts. These are changes in market conditions that can break the cointegration between two assets, meaning their prices no longer move together in a stable way. Hence, this can go against the problem we are trying to solve as high frequency changes in portfolio balancing and risk management to brace through the volatile market can be computationally extensive.

In our project, we tested for pairs that were cointegrated over a one-year period. This allows us to include more pairs, but it also increases the risk of choosing pairs that have high cointegration for a short time. There will always be a tradeoff when selecting timeframes: lower time frame charts offer higher volatility, which can help generate more frequent and responsive trading signals. However, this comes at the cost of increased noise and potential false signals. On the other hand, higher or longer time frames tend to smooth out volatility and price action, providing more stable and reliable signals, but at the expense of fewer trading opportunities and delayed entries.

References

Figueira, M., & Horta, N. (2022). Machine Learning-Based Pairs Trading Strategy with Multivariate. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4295303

Hudson Thames. (n.d.). *Definitive guide to pairs trading*. https://hudsonthames.org/definitive-guide-to-pairs-trading/

Jirapongpan, R., & Phumchusri, N. (2020). Prediction of the profitability of pairs trading strategy using machine learning. *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, 1025–1030. https://doi.org/10.1109/iciea49774.2020.9102013

Krauss, C., Do, X. A., & Huck, N. (2016). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. In Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics, *FAU Discussion Papers in Economics* (No. 03/2016). Friedrich-Alexander-Universität Erlangen-Nürnberg, Institute for Economics. https://www.econstor.eu/bitstream/10419/130166/1/856307327.pdf?ref=https://githubhel p.com

Lence, S., & Falk, B. (2005). Cointegration, market integration, and market efficiency. *Journal of International Money and Finance*, 24(6), 873–890. https://doi.org/10.1016/j.jimonfin.2005.08.001