Applying SVM-Based Trade Classification to Pairs Trading: Statistical Arbitrage

Bowen Zhao, Harsh Kulkarni

Oct 16th, 2025

Motivation & Project Goal

Motivation

- Traditional **pairs trading strategies** rely on simple technical indicators (e.g., RSI, MACD) to detect mean reversion opportunities.
- These methods are often **noisy**, **lagging**, and **not adaptive** to changing market regimes.
- Modern markets are **non-stationary**, with structural breaks and volatility regime shifts → classical signals can degrade quickly.
- By incorporating **econometric spread dynamics**, **macro indicators**, and **Hidden Markov Models (HMM)**, we can **capture richer structure** in spread behavior.
- Machine learning methods like **SVM classifiers** can learn **nonlinear decision boundaries** for trade selection.

Project Goal

- Build a **market-neutral statistical arbitrage strategy** between **cointegrated asset pairs**.
- Use **Support Vector Machine (SVM)** classifiers to predict the profitability of mean-reversion trades.
- Replace simple indicators with a **feature set based on:**
    - **Econometric statistics** (Ornstein–Uhlenbeck process parameters, stationarity tests, Hurst exponent).
    - **HMM regime probabilities** for spread returns.
    - **Macro indicators** (e.g., VIX, yield curve spreads).
- Generate **labels through simulated PnL** of simple z-score mean reversion trades → train two SVMs (long & short).

Data Collection

Select ~10 liquid, same-sector asset pairs (e.g., JPM–BAC, KO–PEP) to ensure comparable fundamentals and stable relationships.

Download adjusted close prices (hourly or daily) and convert to log prices for linearity and variance stabilization.

Align timestamps across assets, handle holidays and missing data carefully.

Collect macroeconomic indicators to enrich the feature space:

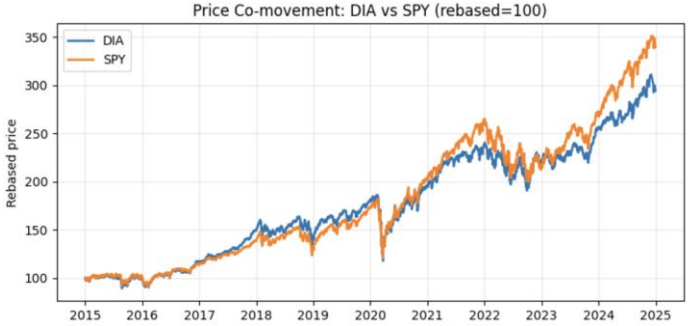
- VIX Index (volatility proxy)
- 10Y–2Y Treasury yield spread (macro regime indicator)

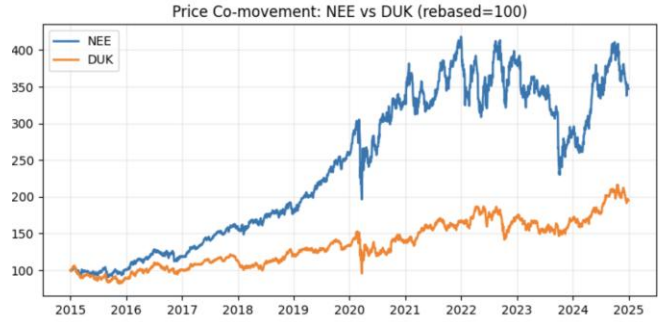
Resample and forward-fill macro data to match the frequency of price data.

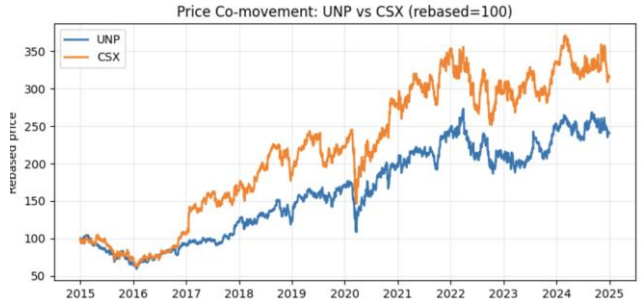
Pair Selection

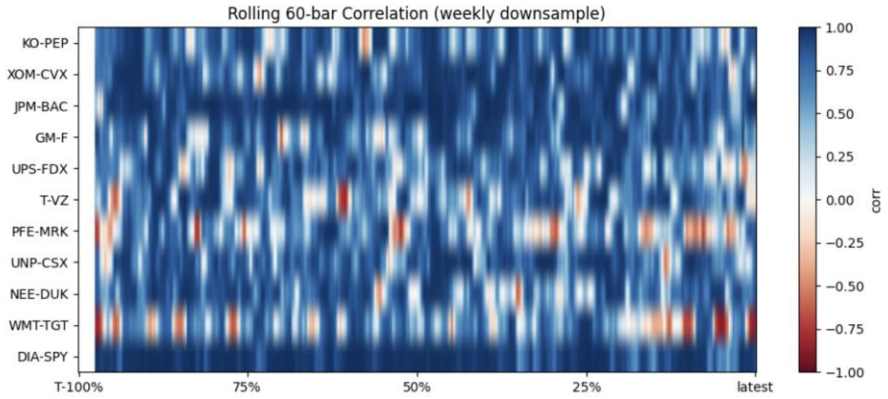
- Apply a **rolling 60-day correlation filter** to identify highly correlated asset pairs (threshold ≥ 0.75).
- Perform the **Engle–Granger cointegration test** on each pair:
  - Regress one asset's log price on the other to obtain residuals.
  - Apply the **ADF test** on residuals; select pairs with p-value < 0.05 to ensure stationarity.
- Recalculate cointegration and stationarity **monthly** to maintain robustness.
- Estimate **Ornstein–Uhlenbeck (OU)** process parameters (κ, μ, σ) for each pair to assess mean reversion characteristics.
- Filter pairs based on OU half-life and stability of parameters.

Pair Selection - Visualization

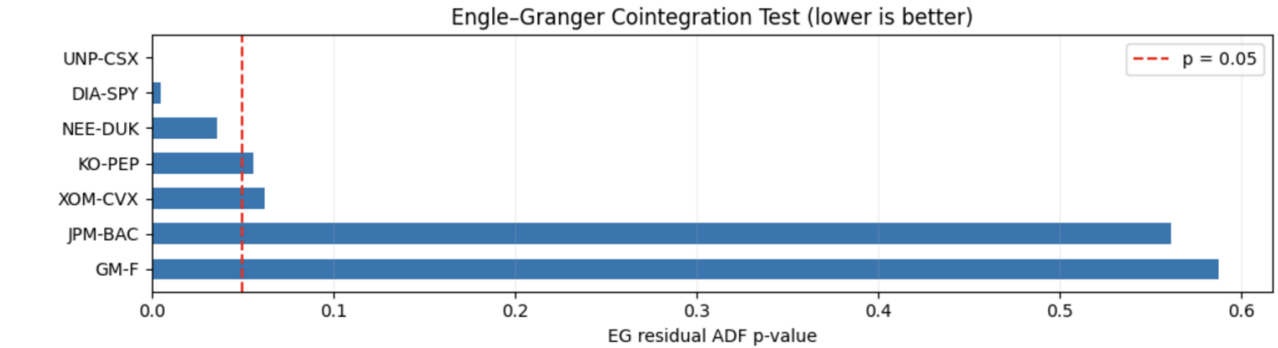


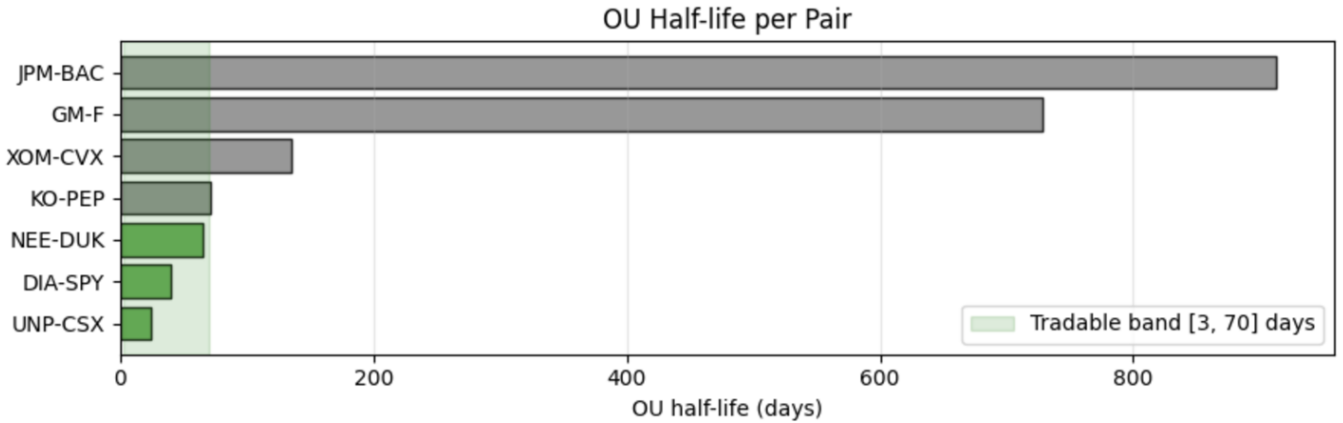






Pair Selection - Visualization





Feature Engineering

Construct predictive features from three categories to capture spread behavior beyond simple technical indicators:

1. **Econometric Features**
    - Estimate parameters of the **Ornstein–Uhlenbeck (OU)** process fitted to the spread: $$dS_t = \kappa(\mu - S_t)dt + \sigma dW_t$$
    - Extract κ (speed of mean reversion), μ (long-term mean), σ (volatility).
    - Additional statistics: ADF t-statistics, Hurst exponent, short-term momentum.
2. **Hidden Markov Model (HMM) Regime Features**
    - Fit a 2–3 state Gaussian HMM to spread returns.
    - Compute **state probabilities** for each time point to characterize regimes (e.g., trending vs mean-reverting)
3. **Macro Features**
    - Include contemporaneous macro indicators (e.g., VIX, yield curve spread) to reflect market conditions.

Label Generation

- Generate **supervised learning labels** by simulating simple **mean reversion trades** on the spread, based on **z-score signals**.
- For each pair, compute the spread and its rolling mean and standard deviation. Then calculate the z-score
- Entry and exit rules:
    - **Enter Long** if $z_t \leq -1.5$
    - **Enter Short** if $z_t \geq +1.5$
    - **Take Profit** at z=0, **Stop Loss** at ±2.5
- For each signal, label:
    - **+1** if the trade hits take profit before stop loss
    - **−1** otherwise
- Separate labels for **long** and **short** signals to train two classifiers (SVM-Long, SVM-Short).

Walk-Forward Data Splitting

Apply a **walk-forward (rolling window)** approach to avoid look-ahead bias and reflect realistic model deployment.

Split the dataset into **training and testing windows** that slide through time:

- **Training Window:** Past $N = 750$ observations (≈ 3 years)
- **Testing Window:** Next $M = 252$ observations (≈ 6–12 months)
- **Label Buffer:** ~30 bars between train and test to prevent label leakage

At each iteration (fold):

- Fit **separate SVM models** for long and short trades using only past data.
- **Calibrate model outputs** to obtain probabilistic scores (*p_profit*).
- Determine a **ROC-optimal probability threshold** on a validation slice within the training window.
- **Evaluate** the calibrated models on the next unseen test window.

Move the window forward and repeat.

This method preserves **temporal ordering**, ensures **no data leakage**, and **mimics live retraining** with dynamic probability calibration per fold.

Backtest Trading Logic

Implement a **live trading simulation** using model predictions and z-score signals.

At each time step:

- Compute spread, z-score, and features (252-bar hedge ratio, 60-bar z-score).
- If **|z| ≥ entry threshold**, raise a signal.
- Side: long if **z < 0**, short if **z > 0**.
- Get calibrated SVM probability for that side, **weighted** by HMM mean-reversion prob.
- Enter, if the effective probability clears the threshold (ROC-optimal per fold/side or top-20% quantile).
- Manage the open trade each bar:
    - Take profit when z -> 0,
    - Stop loss if z moves further past entry,
    - Exit on max holding time.
- **Hyperparameters to tune**: Entry z-score threshold, Take-profit and stop-loss levels, Max holding time

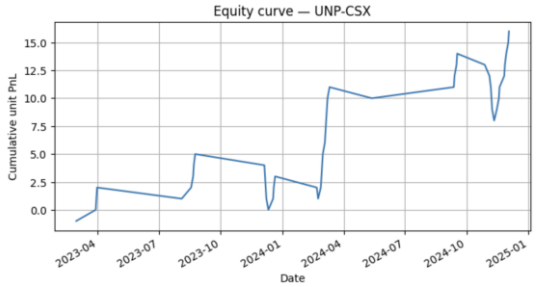
Backtest Results

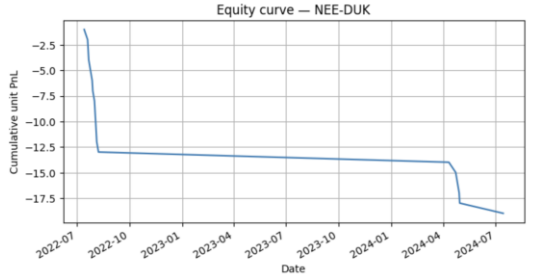
	pair	n_trades	hit_rate	avg_pnl	total_pnl
2	UNP-CSX	48	0.666667	0.333333	16
1	NEE-DUK	19	0.000000	-1.000000	-19
0	DIA-SPY	56	0.285714	-0.428571	-24

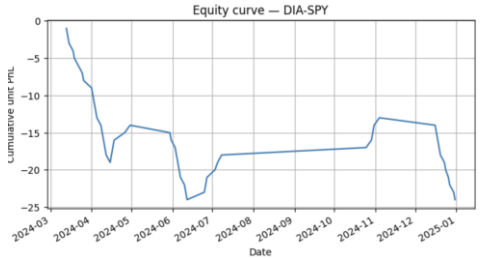
The strategy was evaluated on three final pairs (**DIA–SPY**, **UNP–CSX**, **NEE–DUK**)

Key Findings:

- **UNP–CSX** achieved strong mean-reversion behavior, with a **67% hit rate** and **positive total PnL**.
- **DIA–SPY**, a benchmark pair, traded frequently but underperformed
- **NEE–DUK** exhibited no profitable signals





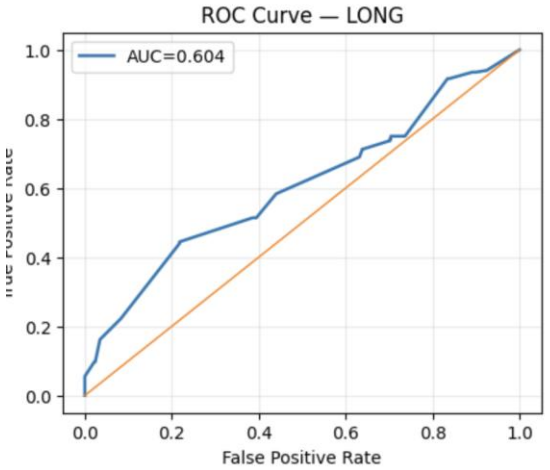


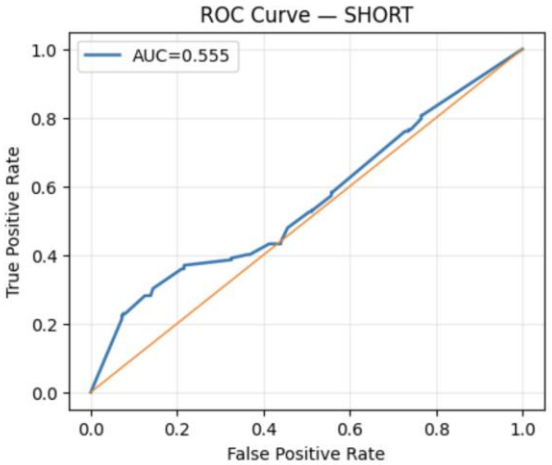
Interpretation

Model behavior shows statistical validity despite low profitability.

- The **ROC curves (AUC ≈ 0.60)** demonstrate that the SVM distinguishes profitable from unprofitable trades better than random, which means it's a **weak but genuine predictive signal**
- The **equity curves** shown earlier evolve smoothly, without erratic jumps, showing the model reacts systematically rather than randomly.
- The **per-pair results** highlight that signal strength varies by market — UNP–CSX mean-reverted successfully, while others lagged due to slower or unstable spreads.

Therefore, even though the total **PnL < 0**, the ROC curve shows us that the model has **directional awareness** of profitable conditions and that the framework works, but execution and parameter tuning need refinement.





Conclusion and Future Work

We,

- Investigated pairs trading performance in different market regimes using a regime-aware SVM framework, integrating econometric features, HMM state probabilities, and back tested trading signals.
- Demonstrated **weak but consistent predictive power** (AUC ≈ 0.6) with our strategy, confirming that mean-reversion structure exists even though total PnL remains modest.
- Implemented Walk-forward design and ROC-based evaluation ensured **no look-ahead bias** and realistic deployment behavior.
- HMM regimes aligned with volatility clustering, validating the market-state interpretation.

Future Work:

- **Hyperparameter Tuning:** Refine z-score thresholds, stop-loss/take-profit bands, and SVM parameters (C, gamma).
- **Regime-Specific Models:** Train separate SVMs for each HMM regime to capture regime-dependent behavior.
- **Execution Layer:** Account for transaction costs and slippage to evaluate real-world performance.