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

Team Member: Bowen Zhao; Harsh Kulkarni

Motivation

Traditional pairs trading strategies often rely on simple technical indicators such as RSI and MACD to detect mean reversion opportunities. However, these tools tend to be noisy, lagging, and insufficiently adaptive to changing market conditions. Modern financial markets are inherently non-stationary, with frequent structural breaks and volatility regime shifts, which can cause classical signals to degrade quickly. This creates a gap between the evolving behavior of asset spreads and the static methods traditionally used to trade them.

To address this, we move beyond simple indicators and incorporate econometric models, macro indicators, and regime-switching techniques. By analyzing spread dynamics more deeply—through statistical processes and market structure—we can capture richer patterns and better anticipate profitable mean reversion opportunities

Project Goal

Traditional pairs trading strategies often rely on simple technical indicators such as RSI and MACD to detect mean reversion opportunities. However, these tools tend to be noisy, lagging, and insufficiently adaptive to changing market conditions. Modern financial markets are inherently non-stationary, with frequent structural breaks and volatility regime shifts, which can cause classical signals to degrade quickly. This creates a gap between the evolving behavior of asset spreads and the static methods traditionally used to trade them.

To address this, we move beyond simple indicators and incorporate econometric models, macro indicators, and regime-switching techniques. By analyzing spread dynamics more deeply—through statistical processes and market structure—we can capture richer patterns and better anticipate profitable mean reversion opportunities

Data Collection

To develop a reliable statistical arbitrage framework, the first step is to build a clean and informative dataset. We select around ten liquid, same-sector asset pairs—for example,

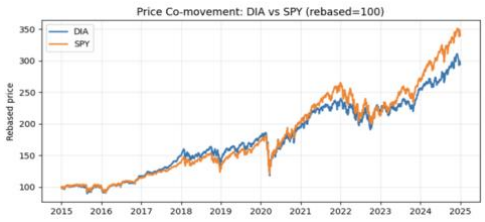
JPM–BAC or KO–PEP—to ensure that both assets share comparable fundamentals and maintain stable long-term relationships. For each asset, we download adjusted close prices at either hourly or daily frequency, convert them to log prices to stabilize variance, and align timestamps carefully to handle holidays and missing data.

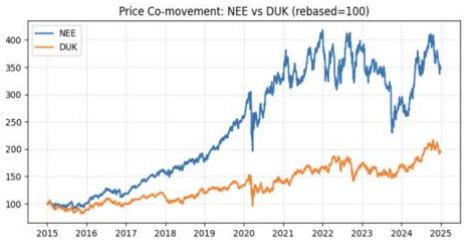
To enhance our feature space, we also collect macroeconomic indicators that provide contextual signals about broader market regimes. Specifically, we incorporate the VIX Index, which serves as a proxy for market volatility, and the 10Y–2Y Treasury yield spread, which captures macroeconomic shifts. We then resample and forward-fill this data so that its frequency matches our price series, ensuring all features are synchronized for downstream modeling

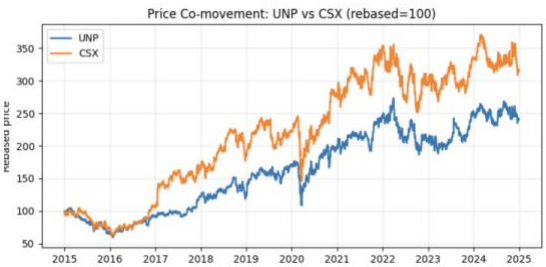
Pair Selection

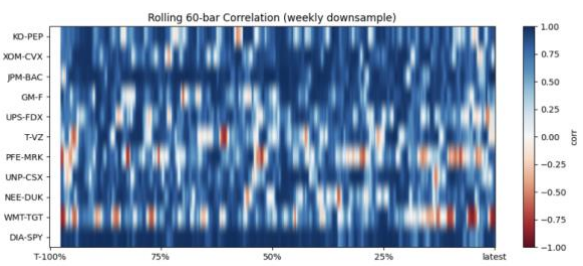
Once the data is collected, the next task is to systematically **identify suitable pairs** for mean reversion trading. We begin with a **rolling 60-day correlation filter**, selecting only those pairs with correlation coefficients above **0.75**. This ensures we're working with assets that historically move together. Next, we apply the **Engle–Granger cointegration test**: we regress one asset's log price on the other to obtain residuals, then use the **ADF (Augmented Dickey–Fuller) test** to confirm stationarity. Only pairs with **p-values below 0.05** are retained, guaranteeing that the spread is statistically mean-reverting.

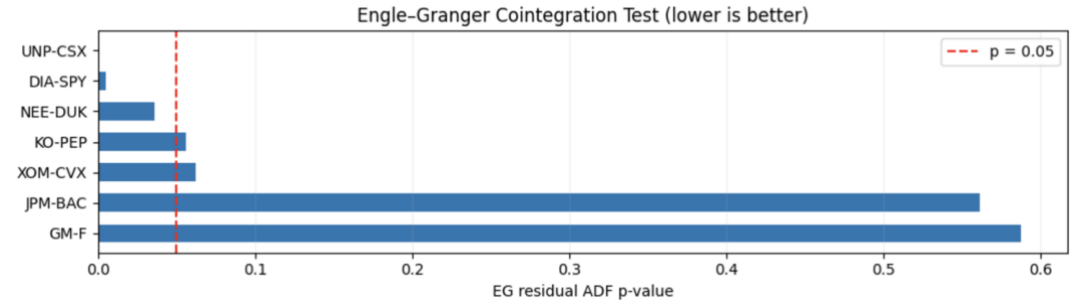
To maintain robustness over time, we **recalculate cointegration and stationarity monthly**, adapting to changing relationships. We also **estimate Ornstein–Uhlenbeck process parameters**—κ, μ, and σ—for each spread, allowing us to assess the **speed and stability of mean reversion**. Pairs with **desirable OU half-lives and stable parameters** make it into the final candidate set

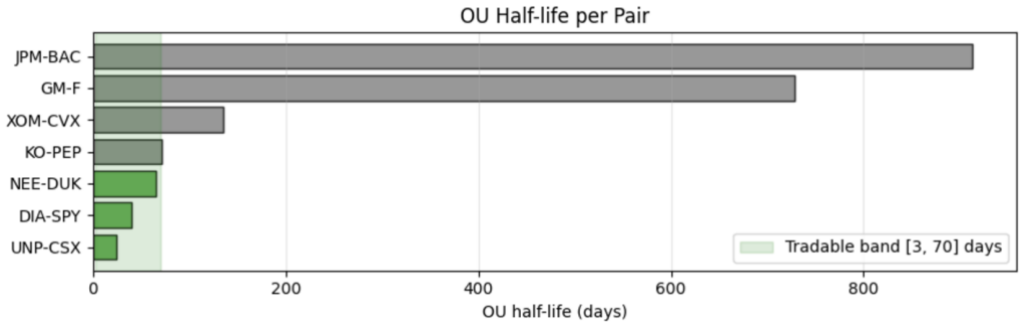












Feature Engineering

To move beyond simple technical indicators, we design a **multi-layered feature set** that captures the **underlying statistical structure** of spreads. Our features are grouped into three main categories:

1. **Econometric Features**
   For each selected asset pair, we fit an **Ornstein–Uhlenbeck (OU)** process to the spread and extract parameters such as **κ** (speed of mean reversion), **μ** (long-term mean), and **σ** (volatility). These parameters help quantify how strongly and how quickly the spread tends to revert to its equilibrium. We also compute complementary statistics like the **ADF t-statistic**, **Hurst exponent**, and short-term momentum, giving the model deeper insight into mean reversion dynamics.

2. **Hidden Markov Model (HMM) Regime Features**
   We fit a **2–3 state Gaussian HMM** to the spread returns to uncover **latent market regimes**—for example, whether the spread is currently in a trending or mean-reverting state. We then compute **state probabilities** at each time point and include them as features, allowing the SVM to adapt its decisions based on the prevailing regime.

3. **Macro Features**
   Finally, we bring in **contemporaneous macroeconomic indicators** such as the VIX index and the yield curve spread. These variables provide contextual

information about market volatility and macro regimes, which can influence the effectiveness of mean-reversion strategies

Label Generation

To train the SVM classifiers, we need a **supervised labeling scheme** that distinguishes profitable trades from unprofitable ones. We generate labels by **simulating simple mean-reversion trades** on the spread using **z-score signals**. Here's how the process works:

- For each pair, we compute the spread, its **rolling mean** and **standard deviation**, and then calculate the **z-score**.
- We define entry and exit rules:
    - **Enter Long** when $z \leqslant -1.5$
    - **Enter Short** when $z \geqslant +1.5$
    - **Take Profit** at $z = 0$
    - **Stop Loss** at $z = \pm 2.5$
- For each trade signal, we assign a label of **+1** if the trade **hits take profit before stop loss**, and **−1** otherwise.

We separate long and short signals, training **two distinct SVM classifiers**—one for long trades and one for short trades. This approach ensures that each model learns the specific patterns associated with profitable entries in each direction

Walk-Forward Data Splitting

To make sure our model evaluation is realistic and avoids **look-ahead bias**, we apply a **walk-forward (rolling window) data splitting** approach. Rather than using a single train–test split, this method **mimics how a live trading strategy would evolve over time**, retraining the model periodically as new data arrives.

We divide the dataset into **training and testing windows** that **slide forward through time**:

- **Training Window:** The past **N = 750 observations** (approximately three years).
- **Testing Window:** The next **M = 252 observations** (about six to twelve months).

- **Label Buffer:** Around **30 bars** between training and testing sets to prevent any label leakage.

For each rolling iteration, we train **two separate SVM models** — one for long trades and one for short trades — using only the **past data**. Then, within the training window, we **calibrate the model outputs** to obtain **probabilistic scores** (p_profit) and determine the **ROC-optimal probability threshold** using a validation slice. This ensures that we don't simply rely on arbitrary cutoffs, but use a threshold that maximizes classification performance in each period.

Finally, we **evaluate the calibrated models** on the next unseen test window, then **move the window forward and repeat**. This **walk-forward design** preserves temporal ordering, ensures **no future information leaks into the past**, and **closely simulates live deployment**, including dynamic probability calibration per fold

Backtest Trading Logic

After training and calibrating our SVM models, we embed them into a **live trading simulation** that mirrors how the strategy would operate in real time. At each time step, we compute the **spread**, **z-score**, and **features** using a **252-bar hedge ratio** and **60-bar z-score** to standardize signals. When the absolute value of the z-score crosses a predefined **entry threshold**, we **raise a trade signal**:

- If **z < 0**, the model considers a **long trade**.
- If **z > 0**, it considers a **short trade**.

For each signal, we obtain the **calibrated SVM probability** for that trade direction and weight it by the **HMM-derived mean-reversion probability**. This ensures that our final decision reflects both **model confidence** and **regime conditions**.

A trade is entered only if the **effective probability exceeds the ROC-optimal threshold** (or belongs to the **top 20% quantile**) determined during model calibration. Once a position is open, we **manage it bar by bar**, exiting under one of three conditions:

1. **Take Profit** when the spread mean-reverts (z-score returns to zero).
2. **Stop Loss** if the z-score moves further past the entry level.
3. **Max Holding Time** to avoid overextending positions.

We also tune key **hyperparameters**, including the **entry z-score**, **take-profit and stop-loss levels**, and **maximum holding time**, to optimize the balance between signal

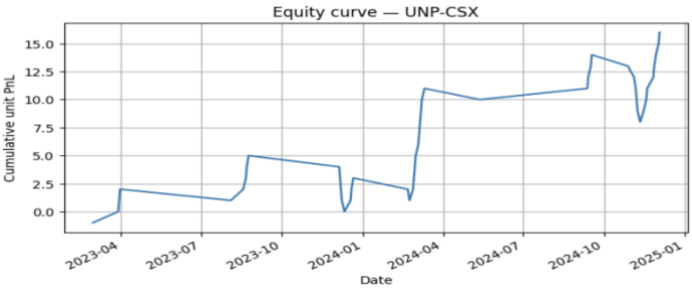
frequency and profitability

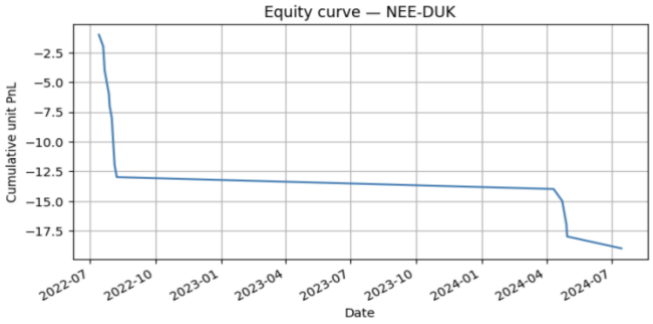
Backtest Results

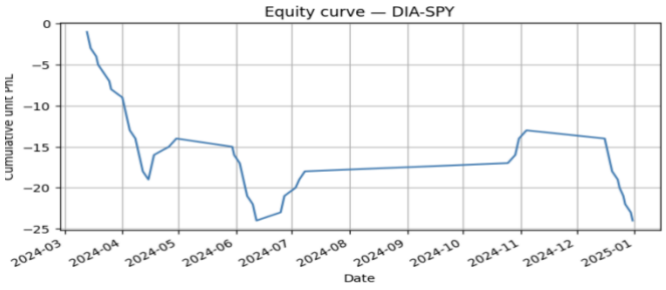
We evaluated the final strategy on three representative pairs: **DIA–SPY**, **UNP–CSX**, and **NEE–DUK**. These pairs were chosen to illustrate different behaviors: DIA–SPY as a benchmark, UNP–CSX for strong mean-reversion properties, and NEE–DUK as a weaker, less profitable pair.

The **UNP–CSX** pair showed the most promising results, with a **67% hit rate** and a **positive total PnL**, reflecting clear and stable mean-reversion behavior. **DIA–SPY** traded frequently but underperformed, highlighting that frequent signals do not necessarily translate into profitability if spreads are too tight or revert too slowly. Meanwhile, **NEE–DUK** generated no profitable signals during the backtest, suggesting the spread dynamics were either too weak or too unstable to support mean reversion

	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





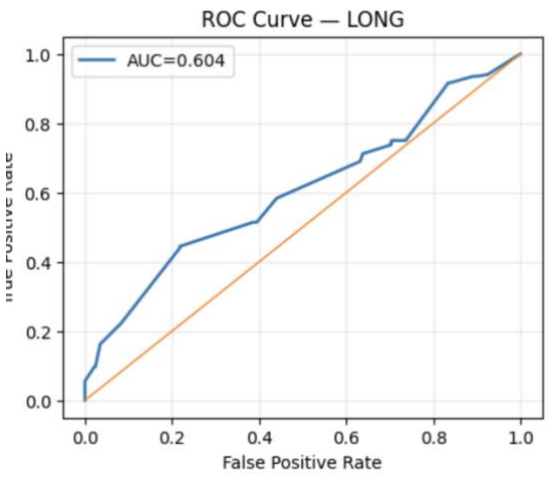


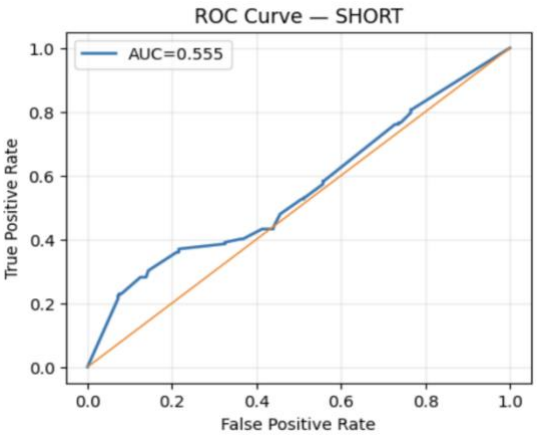
Interpretation of Results

From a modeling perspective, the SVM classifiers achieved **ROC AUC scores of around 0.60**, indicating they can **distinguish profitable from unprofitable trades better than random**, though the signal remains weak. The **equity curves** for the tested pairs evolve smoothly without erratic jumps, demonstrating that the model behaves **systematically**, not by chance.

However, the **total PnL remains below zero**, revealing that while the model has **directional awareness**, **execution parameters** and **threshold tuning** still need refinement. The **per-pair variation** also tells an important story: signal strength

depends heavily on market structure, with some pairs (like UNP–CSX) exhibiting clear mean-reversion patterns, while others do not





Conclusion

In this project, we investigated **pairs trading performance** under different market regimes using a **regime-aware SVM framework**. By integrating **econometric features**, **Hidden Markov Model state probabilities**, and **macro indicators**, we created a richer, more adaptive representation of spread behavior compared to traditional indicator-based methods.

Our results showed **weak but consistent predictive power** — with ROC AUC scores around **0.6** — confirming that meaningful mean-reversion structure exists in the data. Importantly, our **walk-forward design** and **ROC-based thresholding** ensured that model evaluation was free from **look-ahead bias**, closely resembling real-world deployment. Additionally, the alignment between **HMM regimes** and **volatility clustering** validated our approach to capturing market states

Future Work

While the framework demonstrates statistical validity, there's significant room for **performance improvement**. Future steps include:

- **Hyperparameter Tuning**: Refining z-score thresholds, stop-loss/take-profit levels, and SVM parameters (e.g., C and gamma) to improve signal precision.
- **Regime-Specific Models**: Training separate SVMs for each **HMM regime**, allowing the model to specialize in different market environments.
- **Execution Layer Enhancements**: Incorporating **transaction costs**, **slippage**, and **latency effects** to evaluate real-world performance more accurately.

By focusing on these areas, the strategy can move from a **statistically valid but modest** signal toward **a practical, profitable trading framework**