Pairs Trading With GMM-MinHash Regime Signaling

CSCI 4022
Advanced Data Science

Dylan Spektor
Dylan.Spektor@colorado.edu — CSCI & APPM

Under the guidance of
Professor Osita Onyejekwe

College of Engineering and Applied Science
University of Colorado Boulder
April 30th, 2026

Contents

Introduction

Pairs trading is a well-established quantitative investment strategy and perhaps the most prominent within the broader category of statistical arbitrage. Strategies within this category aim to identify inefficient pricing between various equities through statistical methods (Avellaneda & Lee, 2010).

In quantitative finance, inefficient pricing refers to asset pricing that is believed to deviate from a theoretical fair value due to some failure to consider all available market information or dynamics. The motivation for identifying inefficiencies is the ability to profit from correcting them, e.g. buying an asset whose current price is below fair value or, inversely, shorting an asset that is overpriced. This view of the market is derived from the foundational Efficient Market Hypothesis, which, for the purposes of this report, can be crudely interpreted to support the strategy of pairs trading as a method of correcting market prices to reflect available information (Fama, 1970).

As may be inferred from its name, pairs trading specifically applies this market lens to pairs of assets, scrutinizing their relative prices to identify profitable corrections. More precisely, the historical spreads, or differences, between the prices of two assets are statistically analyzed to determine whether the prices tend to move together, or, after temporary instabilities, converge back to some equilibrium (Gatev et al., 2006). As this report hopes to demonstrate, pairs trading inspires sophisticated and highly effective implementations despite its remarkably simple conceptual foundation.

A canonical tool applied to the pairs trading problem is the Engle-Granger cointegration test, which assesses whether two asset prices share a long-run relationship, or stable equilibrium, despite individual fluctuations (Engle & Granger, 1987). This report applies the cointegration test along with several course methods.

Course methods applied in this report include dimensionality reduction, PageRank, Gaussian Mixture Modeling (GMM), and MinHashing. The latter two are used in conjunction to first identify regimes of pair spreads arising from market dynamics such as volatility, and then using these classifications along with comparisons to historical windows in order to generate short, neutral, and long signals corresponding to changes in the spread. Additionally, a simple supervised model layer is built on top of this signaling apparatus to identify possible improvements in the resulting PnL (Profit and Loss) and other metrics of the strategy.

In performing this procedure, we seek to understand whether pricing inefficiencies can be reliably detected with the proposed GMM-MinHash signaling apparatus. We ultimately hope to answer whether the proposed procedure is a profitable investment strategy, in turn exploring more fundamental questions regarding the behaviors of pair spreads in equity markets.

Methodology

A fully parameterized pipeline for the entire procedure is given in `07-full-pipeline.ipynb`.

Data

Our strategy is built on equities within the S&P 500 as of April 2026. A list of these tickers was downloaded from a dataset publicly available on Github[1].

Price data for these tickers was collected using the `yfinance` Python library. Note that this library is an unofficial scraper of the Yahoo Finance website, so data may contain inconsistencies. However, this method is appropriate for the scope of this project.

For each asset for which data could be scraped (497/500 assets), daily closing price was collected every day from January 1st, 2016 to January 1st, 2026. The first 8 years of data will be used as a training set, whereas the last two years are split evenly into the validation and testing sets, chronologically. The actual training set only begins on July 6th, 2016, likely due to difficulty scraping data from further back.

The data is cleaned with standard methods, with any ticker missing data for more than 5% of trading days being dropped. For those missing a proportion of data under this threshold, missing prices are forward-filled using the previous day's closing price. Finally, the data is log-transformed as follows:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

where r_t is the log return for day t and P_t and P_{t-1} are the returns for the current day and previous day, respectively. Log-transforming returns in this manner is a common practice in quantitative finance due to the convenient property of being able to simply add log returns to track compounding returns rather than multiplying.

The next step is to perform dimension reduction via PCA. We standardize the prices in each column to a zero mean and standard variance, taking care to only derive the transformation from the training set, then apply it to all sets. This treatment will be the norm in this project to avoid look-ahead bias, i.e. using future information from the validation and testing sets considered to be out-of-sample. This is so that the simulated returns of our strategy remain realistic.

[1] https://github.com/datasets/s-and-p-500-companies/blob/main/data/constituents.csv

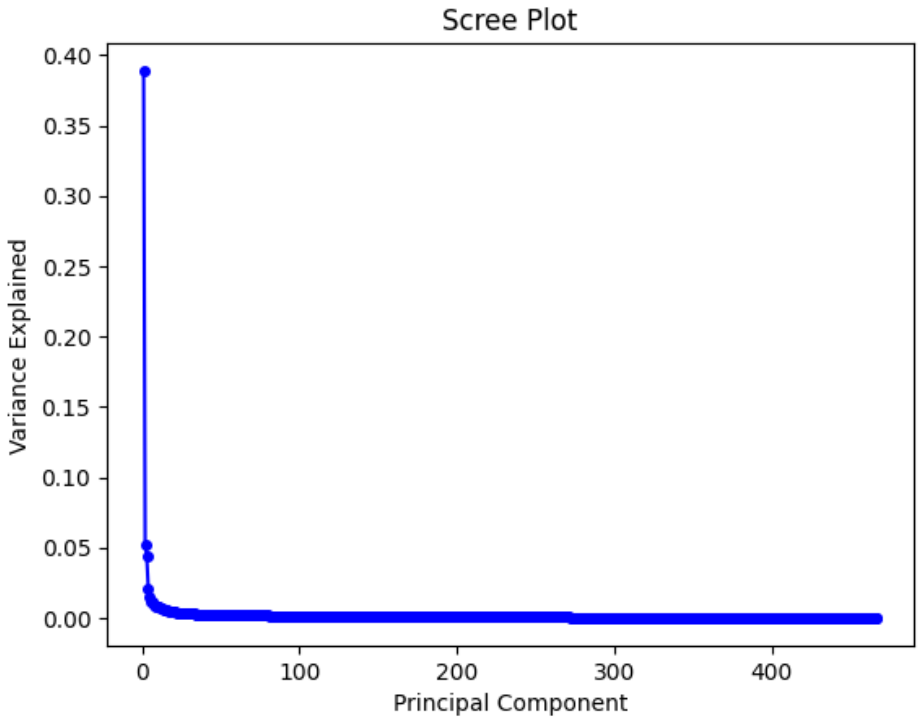

Figure 1: Scree plot produced by PCA on log equity returns.

Figure 1 demonstrates that the equity returns data is highly correlated, as shown by the very steep and early drop in variance explained. Due to this, we find it sufficient to keep the first ten components, which cumulatively explain roughly 57% of variance. PCA is fit on the training set and then all sets are transformed, with the resulting residuals being stored as csv files.

All data, raw and manipulated, is accessible in the Github repository for this project.

Pair Selection

This step of our procedure filters our asset universe with PageRank and applies the Engle-Granger cointegration test to identify highly correlated pairs for which pricing inefficiencies may exist.

First, a Pearson correlation matrix is generated for the training residuals using `pandas`, producing Figure 2.

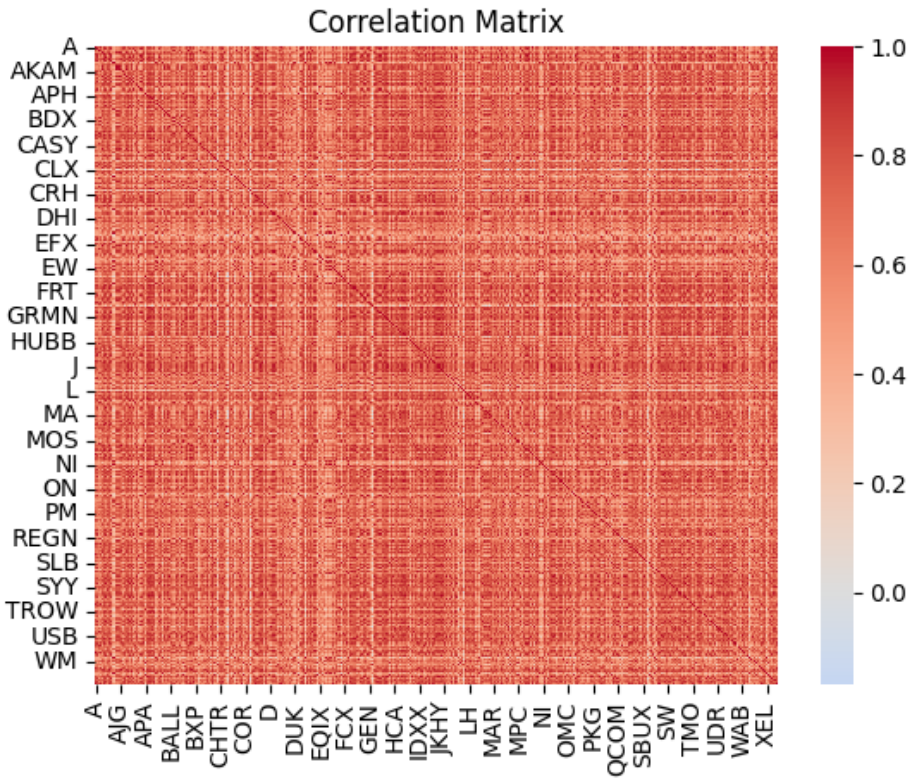

Figure 2: Correlation matrix heatmap.

This heatmap confirms our expectations that most equities are highly correlated.

Purely heuristically, we select a correlation threshold of 0.8 to filter only the most correlated equities out of an inherently highly correlated set. This threshold was also partially determined by observing the binary representations in Figure 3 and choosing a threshold such that the diagonal line representing the trivial 1.0 correlations between each asset and itself is distinctly visible against the rest of the plot.

Next, we seek to apply weighted PageRank as another filter to keep the top 25 assets with the highest scores. The weight matrix is simply a copy of the correlation matrix with all weights below the chosen threshold of 0.8 set to zero and the rest squared to place emphasis on stronger correlations. The PageRank algorithm used is a standard implementation that normalizes the rows of our weight matrix to ensure that the typical probabilistic interpretation of PageRank holds.

The relatively uniform score distribution across the top assets observed in Figure 4 suggests that this filter may be capturing mostly a single common market factor rather than individual relationships. This is despite PCA being implemented to prevent this exact effect. However, this is acceptable given that the purpose of PageRank in our procedure is only to choose an acceptable subset of assets to be permuted to create $P(25, 2) = 300$ candidate asset pairs. This coarse reduction of the universe is necessary to avoid feeding an intractable number of candidate pairs into the rest of the procedure.

We now apply the Engle-Granger cointegration test on our PageRank-selected asset pairs by applying the test to each pair using the `stats.models.tsa.stattools.coint` imple-

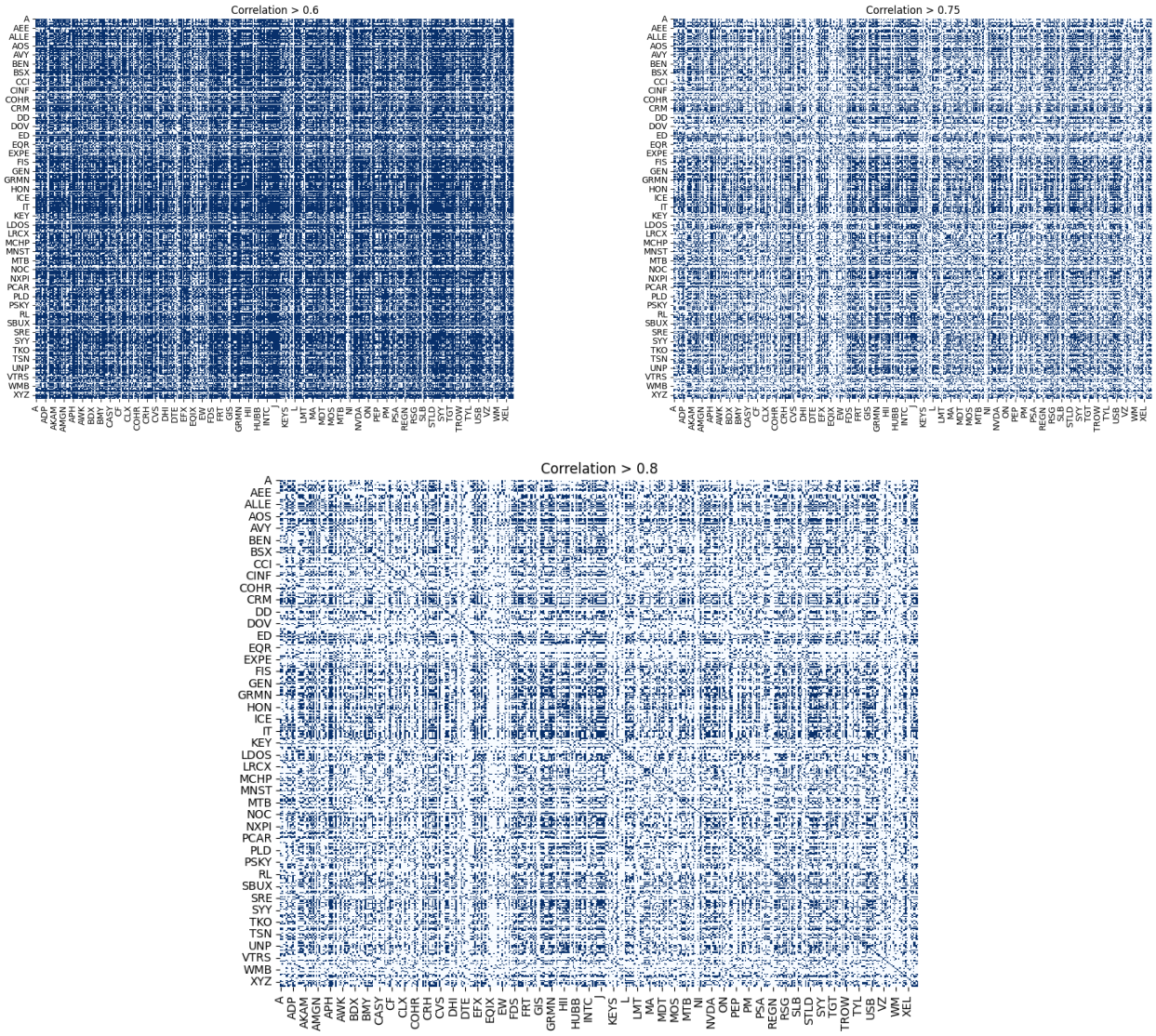

Figure 3: Binary representations of correlation matrix filtered for pairs above given threshold.

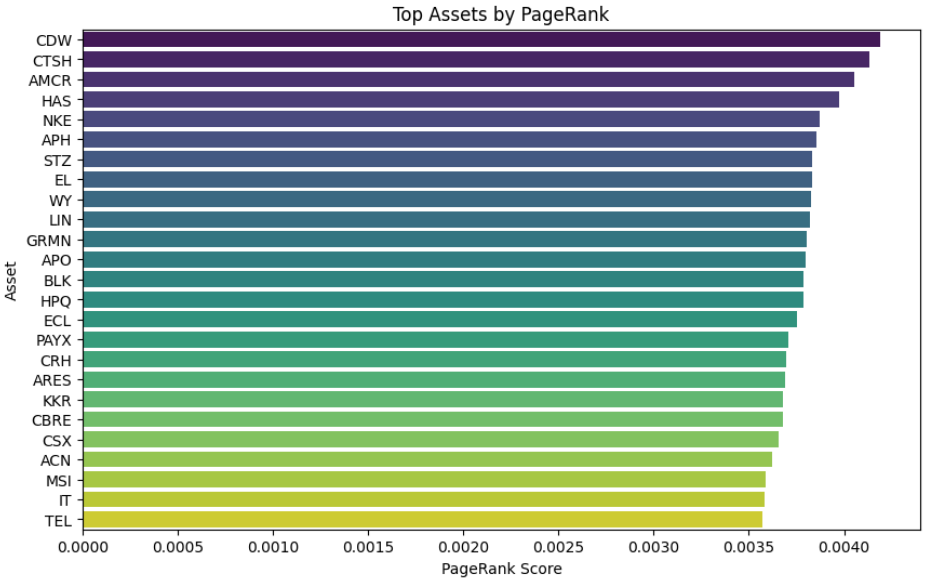

Figure 4: Top 25 weighted PageRank scored assets.

mentation. In accordance with standard practice, we only consider pairs for which the test returns a p-value lower than a threshold of 0.05 to be cointegrated. For each pair, we linearly regress each asset's spread series against each other to compute β, a constant known as a hedge ratio that tells us which linear combination of the two assets' price series yields the closest to a stationary time series. This effectively tells us that buying a share of one asset is equivalent to buying β shares of the other asset. Note that the order the assets appear in each pair is meaningless.

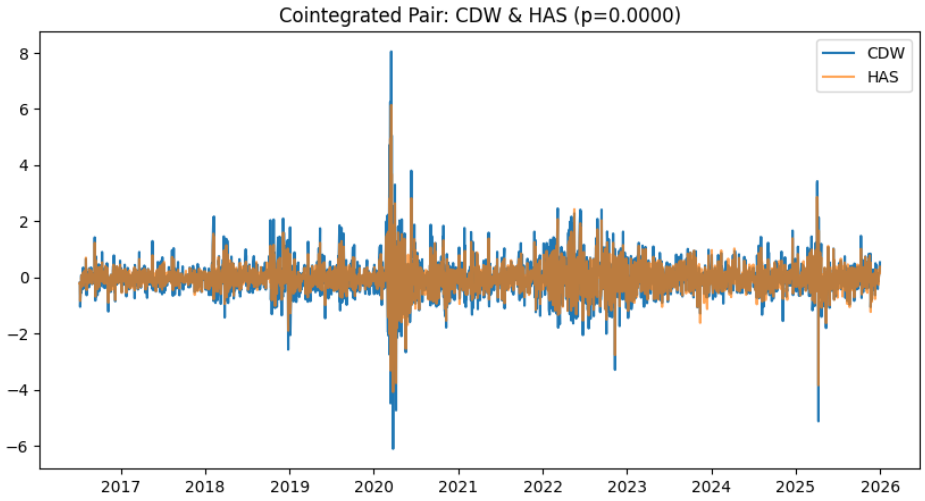

Figure 5: Cointegrated pair CDW-HAS.

As observed in Figure 5, we are immediately able to see highly cointegrated movements between asset pairs such as CDW and HAS, for which the p-value gets rounded off to zero.

Now that we have a set of pairs which we are confident exhibit exploitable pricing inefficiencies, we move on to classifying their spreads into regimes and signaling trades.

Regime Classification and Signaling

In this section we use GMM in conjuction with MinHash to construct a robust trade signaling apparatus. Note that all fitting is done on training data to avoid look-ahead bias.

GMM

We fit a Gaussian mixture model to each spread using `sklearn.mixture.GaussianMixture`. We seek to be able to observe the spread between two assets and immediately classify it within a specific regime arising from the GMM. In financial terms, these regimes are proxies for market factors affecting trades of the pair's assets, the primary of which we are concerned with is volatility.

The number of regimes we impose, i.e. k, is chosen by minimizing the Bayesian Information Criterion (BIC) for the range $k \in \{2, 3, 4, 5\}$, which was chosen to allow freedom in classifications while reasonably expecting the spread to only exhibit a few regimes.

We now seek to use these regimes to inform trades. As a risk-averse practice, we choose to only trade within the regime classified as the calm regime, i.e. the regime with the lowest standard deviation. This is to avoid trading in highly volatile regimes where there is a greater chance of a trade resulting in a loss.

Simultaneously, our goal is to predict large price corrections. We target spreads that are most extreme within the calm regime so that there still exists a predictable significant movement in the price. Anomalous spreads are detected using the z-score, a common measure we define as:

$$z = \frac{s - \mu_{\text{spread}}}{\sigma_{\text{spread}}}$$

for a given spread value s. This measures how many standard deviations s is from the mean.

We compute our z-scores in relation to the entire spread data's mean and standard deviation, and choose the following thresholds for signals:

Listing 1: z-score signaling.

```
# "long" (buy)
signal[(z_scores < -1) & (sorted_labels == calm_regime)] =  1
# "short" (sell)
signal[(z_scores >  1) & (sorted_labels == calm_regime)] = -1
# "hold" (neutral)
signal[z_scores.abs() < 0.2] = 0
```

The resulting distribution of trade signals for each pair disproportionately consists of neutral "hold" signals. However, these are the thresholds derived from tuning until a significant number of "short" and "long", i.e. trade, signals appeared for most pairs. In other words, we seek to balance identifying the maximum number of pricing inefficiencies with only trading when we are confident of a reversion. To ensure we are only trading active pairs where our

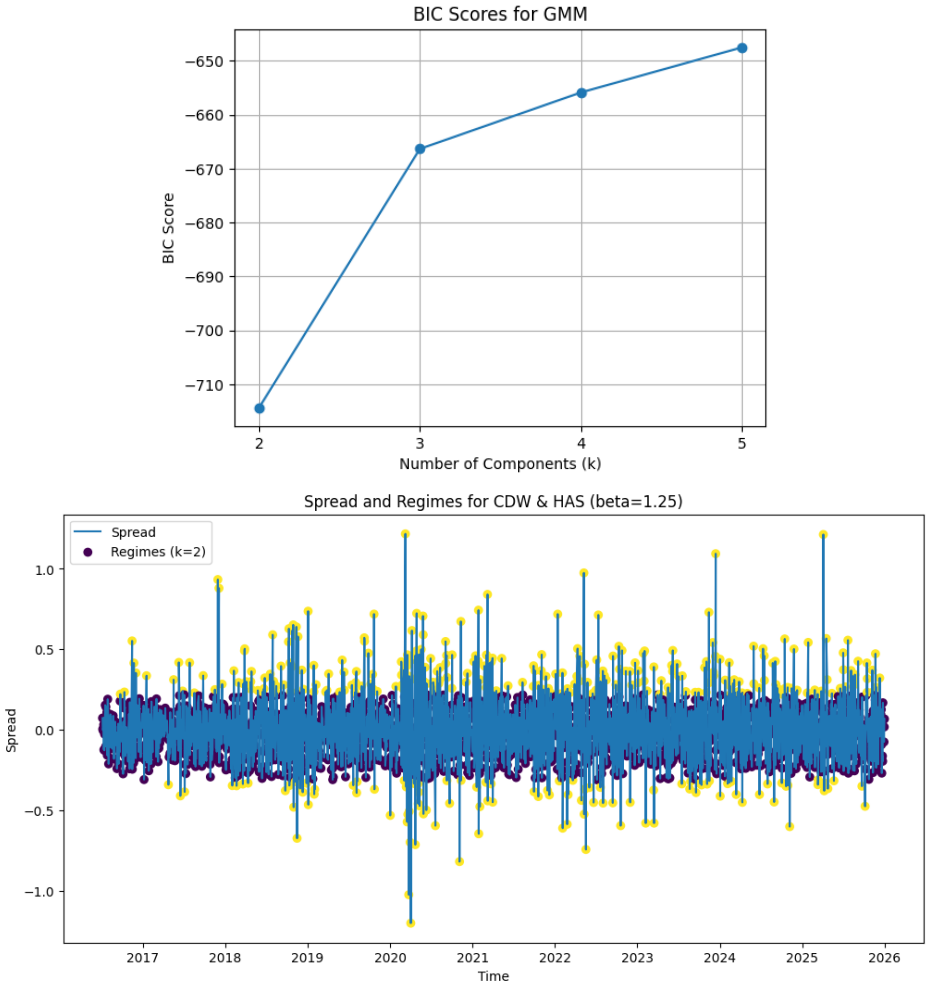

Figure 6: BIC selection process and dual regime classification for CDW-HAS.

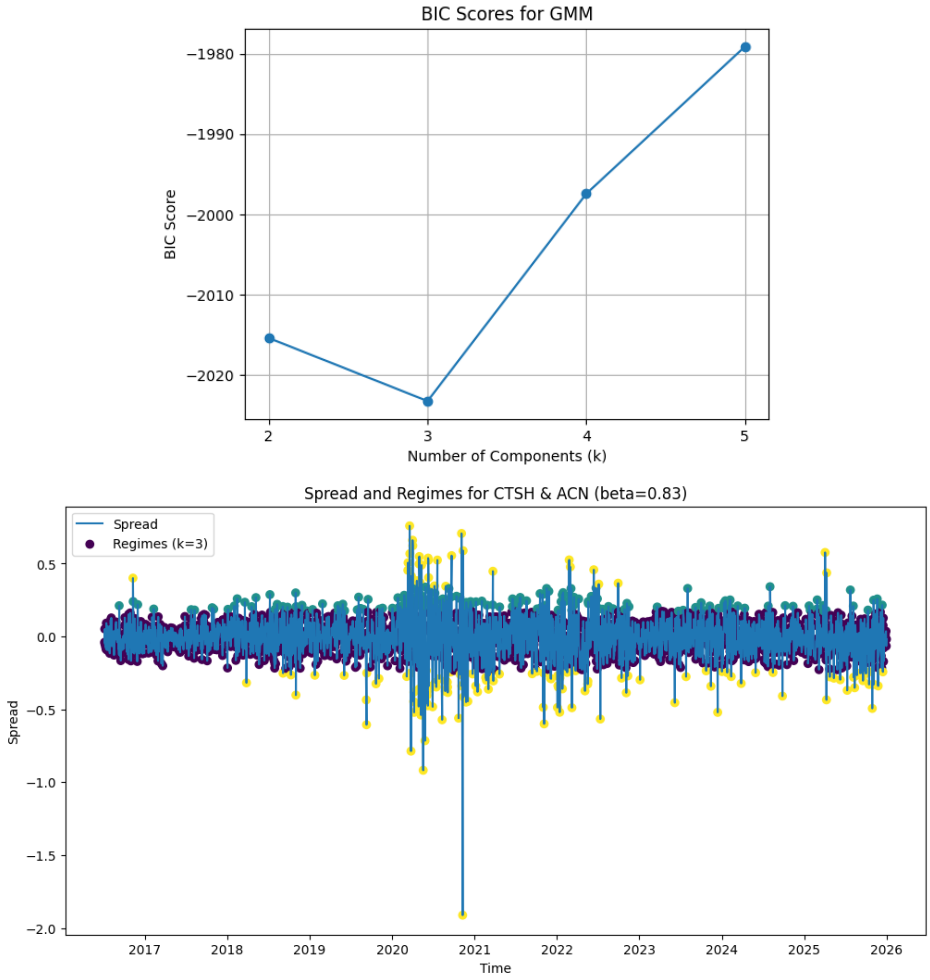

Figure 7: BIC selection process and triple regime classification for CTSH-ACN.

capital will not become stuck in between sparse trades, we further filter the pairs by those with a minimum of 200 GMM-produced trade signals before moving on to the following MinHash procedure.

MinHash

The purpose of our MinHash implementation is to similarly produce signals for trades. However, these signals are based on historical performance, aiming to supplement the regime-based signaling of GMM with pattern-matching.

Specifically, we sort the global z-scores computed for each spread into five discrete categories represented by symbols:

Listing 2: Discretization of spread.

```
z     = (spread_window - mean) / std

symbols = []
for val in z:
    if val < -1.0:        symbols.append('L')
    elif val < -0.2:      symbols.append('l')
    elif val <  0.2:      symbols.append('0')
    elif val <  1.0:      symbols.append('h')
    else:                 symbols.append('H')
```

We then produce a $k = 3$ shingle set, with this relatively small k being chosen to ensure that our signaling will be highly sensitive to very small differences between historical spread windows. We MinHash the shingle set using the `datasketch` implementation.

The crucial step is building our LSH index. We choose a window size of 20 days, meaning the last 20 days of data relative to a given spread. We apply this rolling window to the entire training set to build the index. To determine a signal for a spread, we then look at the rolling window ending at that spread and query the LSH index for similar historical windows. Then, for each historical window, we compute the 10-day forward return, i.e. the change in the spread over the next ten days. Our MinHash signal thus becomes the averaged forward returns of all historical windows. If no similar historical windows exist, the signal becomes an inconclusive zero. Note that for values near the end of of the training set time period, the forward return may leak into data from the validation set. However, this produces minimal bias and will not affect the evaluation of the final model.

Finally, we are able to combine our GMM and MinHash signals into a comprehensive final signal as follows.

Listing 3: Final signal combination.

```
def combine_signals(gmm_signal, minhash_signal):
    if gmm_signal == 0:
        return 0
    # confirm if minhash agrees in direction, suppress if it disagrees
    if gmm_signal == 1 and minhash_signal > 0:
        return 1
```

```
if gmm_signal == -1 and minhash_signal < 0:
    return -1
return 0  # disagreement, so stay flat
```

Observe that we disregard the magnitude of the MinHash signal and simply compare the signs of both signals, with the final signal representing either the inconclusiveness of their disagreement or the direction of their agreement.

Signaling Analysis

We now briefly analyze the frequency of agreements and disagreements between the GMM and MinHash signals to better understand the effectiveness of the final signaling apparatus.

We define the regime-first signal match percentage as

$$P(\text{GMM and MinHash signals match} \mid \texttt{gmm\_signal} \neq 0)$$

and the MinHash-first signal match percentage as

$$P(\text{GMM and MinHash signals match} \mid \texttt{minhash\_signal} \neq 0).$$

We are essentially analyzing the frequency of agreements between the two signals given a trade signal on either side.

The results are given in Figure 8. We observe that there is little difference between the distributions for regime-first and MinHash-first match percentages. However, we observe a clear decrease in match percentage on the validation and testing splits as is expected given the signaling was derived from the training set. In fact, this is a convincing result to show that results are not significantly skewed by look-ahead bias. We observe the validation and testing sets experiencing match percentages around 50%, i.e. the signals agree half of the time. We believe this average to be acceptable and in fact necessary to exclusively consider trades we are confident will be profitable. Moreover, the fact that the average match percentage between validation and test sets remains roughly constant suggests that the signaling apparatus is robust to change in pricing trends over time for a period of at least two years.

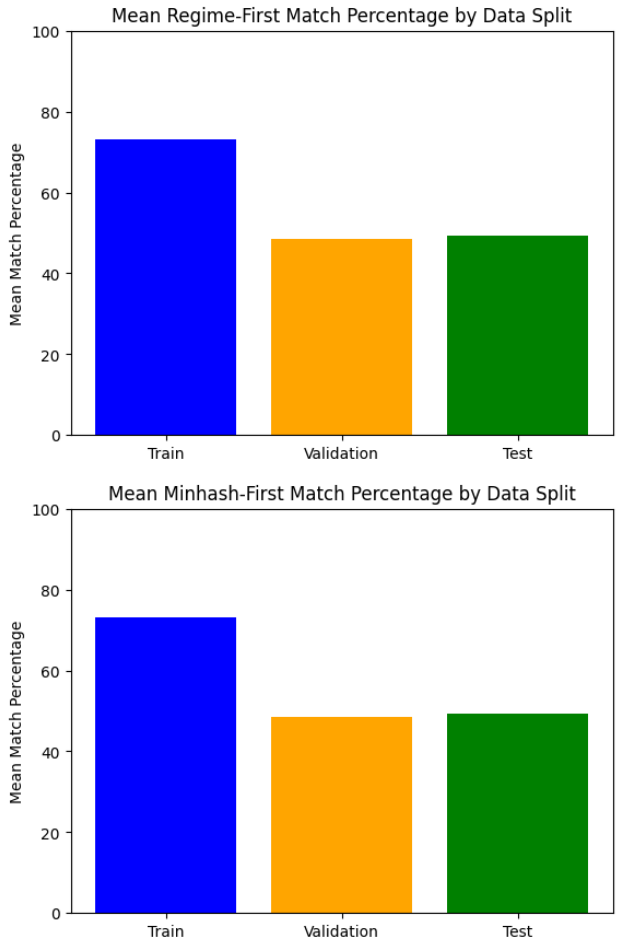

Figure 8: Signal match percentages.

Now that we are confident in the effectiveness of our final signal, we complete our signaling apparatus by training a supervised model on the signal.

Supervised Model

We explore two types of supervised models: logistic regression and a random forest classifier. These are implemented using the modules `sklearn.linear_model` and `sklearn.ensemble`, respectively. We train them to predict the final combined signal.

Listing 4: Model features and target.

```
FEATURES = ["spread", "z_score", "regime"]
TARGET = "final_signal"
```

Our models are instantiated as follows:

Listing 5: Logistic regression instantiation.

```
model = LogisticRegression(
    solver='lbfgs',
```

```
    class_weight='balanced',    # handles 0-heavy imbalance
    max_iter=1000,
    random_state=42,
)
```

Listing 6: Random forest classifier instantiation.

```
model = RandomForestClassifier(
    n_estimators=100,
    max_depth=4,
    random_state=42
)
```

As has been practiced throughout this report, we ensure the models are fit only on the training data. We analyze the resulting model using `sklearn.metrics.classification_report`, which allows us to see each model's precision, recall, and F1-score per signal class. Precision is the fraction of all predictions made for the class that were correct. Recall is the fraction of signal instances found out of all actual instances of the signal. The F1-score is the harmonic mean of precision and recall, providing a single score between 0 and 1.

Below is the report for the linear regression model on the training data:

Class	Precision	Recall	F1-score
Short (-1)	0.32	1.00	0.48
Exit (0)	1.00	0.74	0.85
Long (1)	0.37	1.00	0.54

Table 1: Logistic regression training performance.

We observe that precision is poor for short and long signals, indicating that this model handles the class imbalance poorly, with only a third of its short and long predictions being correct.

Below is the report for the random forest classifier (RFC) model on training data:

Class	Precision	Recall	F1-score
Short (-1)	0.84	0.60	0.70
Exit (0)	0.97	0.98	0.97
Long (1)	0.81	0.88	0.84

Table 2: Random forest classifier training performance.

For this model, we observe much higher precision and a more robust response to the class imbalance, with the lowest F1-score being an acceptable 0.70 for the short signal.

This improved performance is observed for the validation set as well, whose report is mostly equivalent, and in fact experiences minor improvements for some metrics.

As such, we will use the RFC to produce a `model_signal` variable with support $\{-1, 0, 1\}$, following the convention of `final_signal`. The effectiveness of this signal for informing trades will be compared to the GMM-MinHash signal in our backtest.

Class	Precision	Recall	F1-score
Short (-1)	0.86	0.60	0.71
Exit (0)	0.97	0.98	0.97
Long (1)	0.81	0.89	0.85

Table 3: Random forest classifier validation performance.

Results

In this section, we explore the results of a backtest of our strategy. In quantitative finance, a backtest involves applying a strategy to historical data to determine whether it would generate a profit had it been run. We backtest on all splits, including the test split. Thus, this backtest is intended as a final result given we are pleased with the tuning of all hyper-parameters.

The backtest measures the success of our strategy in terms of PnL (Profit and Loss), the most fundamental metric which simply demonstrates the returns of simulated trades. For the sake of simplicity, we will only be making trades on the first asset of each spread, buying it in response to long signals and shorting it in response to short signals. Shorting the asset is simulated by simply adding the negative of the spread's difference between subsequent days to our returns. Recall that we determine the amount of shares to buy using the hedge factor β. Note that potential transaction costs in actual markets are neglected in this simulation. Our figures represent averaged results over all traded pairs.

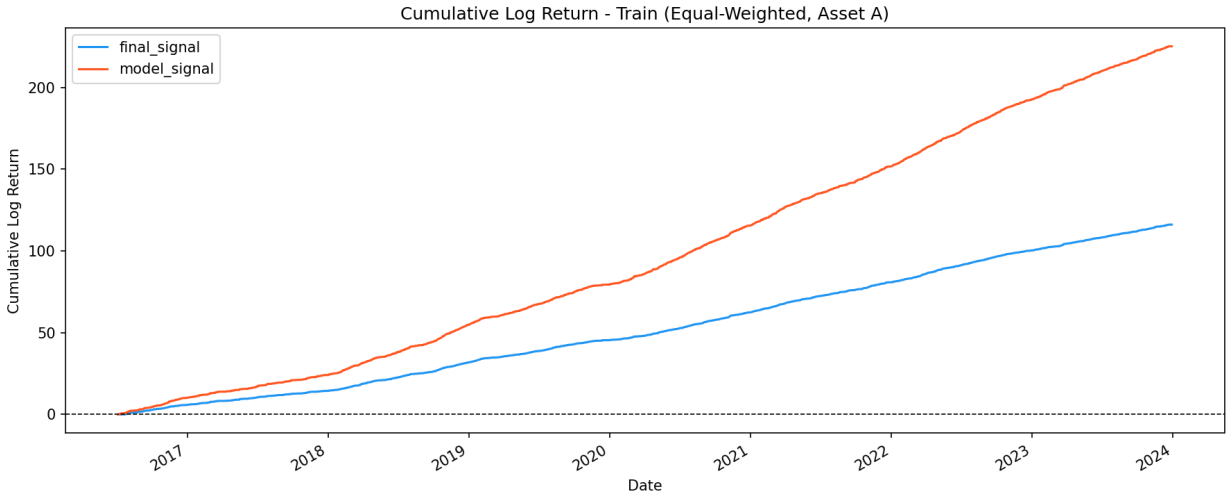

Figure 9: PnL of the training backtest.

From Figure 9, we immediately observe that:

- Our trades using both the GMM-MinHash signal and the RFC signal appear to produce steady and significant positive returns with remarkable stability.
- Trades informed by the model signal appear to generate significantly more profit than those informed by the GMM-MinHash signal, with the final cumulative log return

of the model-informed trades of about 200 being roughly double that of the GMM-MinHash-informed trades.

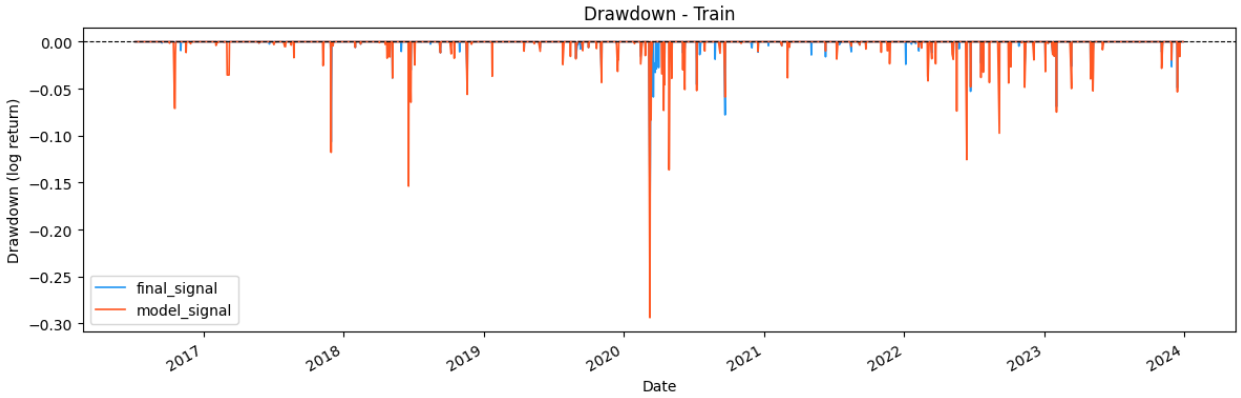

Figure 10: Drawdown plot of trades made on training data.

The stability and magnitude of our returns indicate that the strategy was successful, but this backtest is highly limited and based only on simulations, the integrity of which should be questioned. We will discuss limitations further in the Conclusion section. Furthermore, the second observation is initially unexpected, as the model is meant to predict the final signal and is therefore anticipated to have similar or worse performance. We suggest excessive final signal exclusivity as a potential explanation. Specifically, by imposing the requirement of agreement between GMM and MinHash signals to produce the final signal, we make the final signal highly safe and exclusive, likely not producing signals in many cases where a profitable trade could be made. The RFC model may successfully generalize our final signals such that we take on many more trades than we would if only using the final signal. Indeed, in 10 we see that the model-informed trades experience higher drawdown, i.e. decline from a peak, resulting in a more volatile but higher-return indicator.

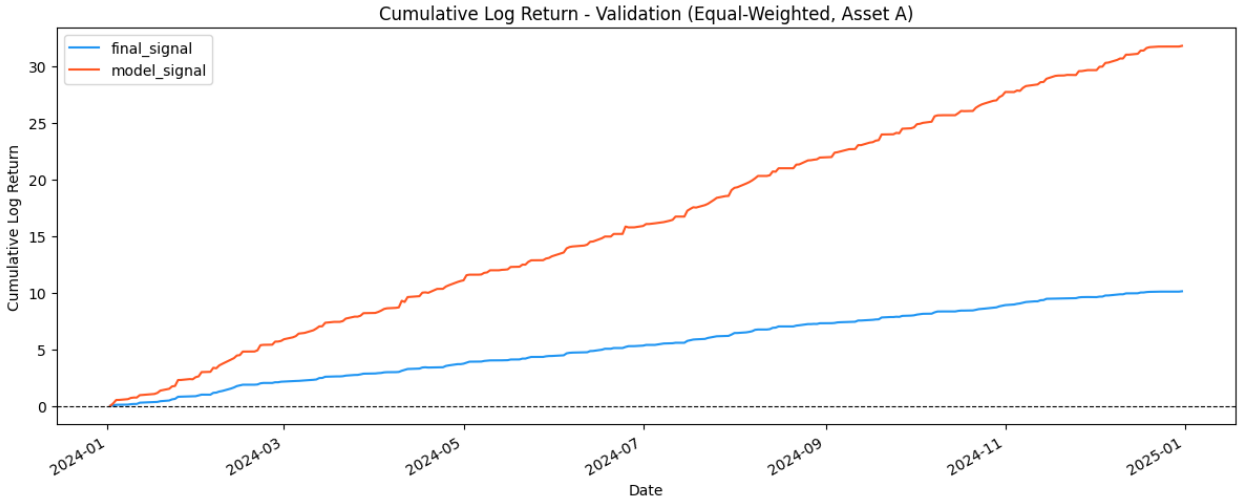

Figure 11: PnL of the validation backtest.

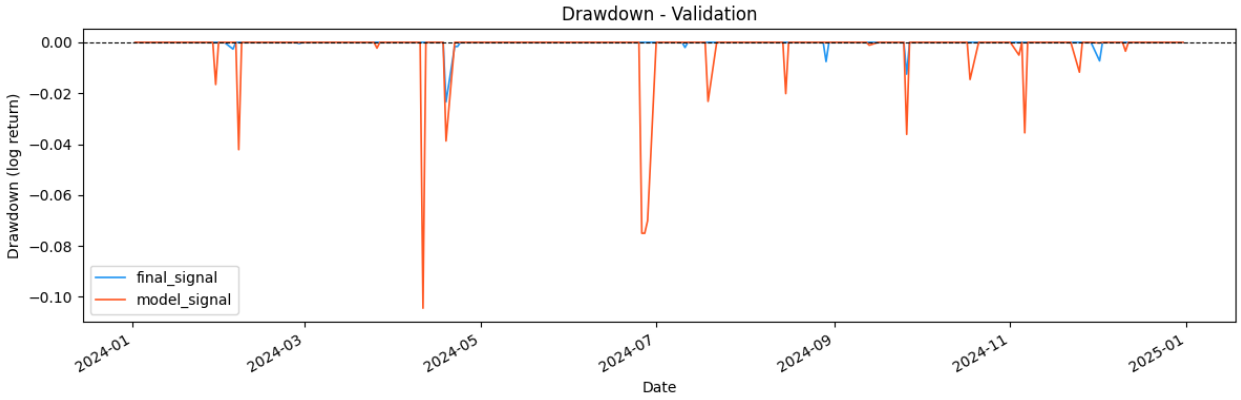

Figure 12: Drawdown plot of trades made on validation data.

We observe from the backtest done on the validation period in Figures 11 and 12 that these results carry over to out-of-sample data, except now the model-informed trades outperform the GMM-MinHash-informed trades by a factor of almost 6, indicating that the previously observed effect is greatly exacerbated out-of-sample. Moreover, we see that the cumulative log return by the end of the period is roughly a third of that experienced over the training period. Given that the training period is eight years and the validation period is one year, this represents a much higher growth rate and therefore heightens suspicions that our backtest may be inflating expected PnL.

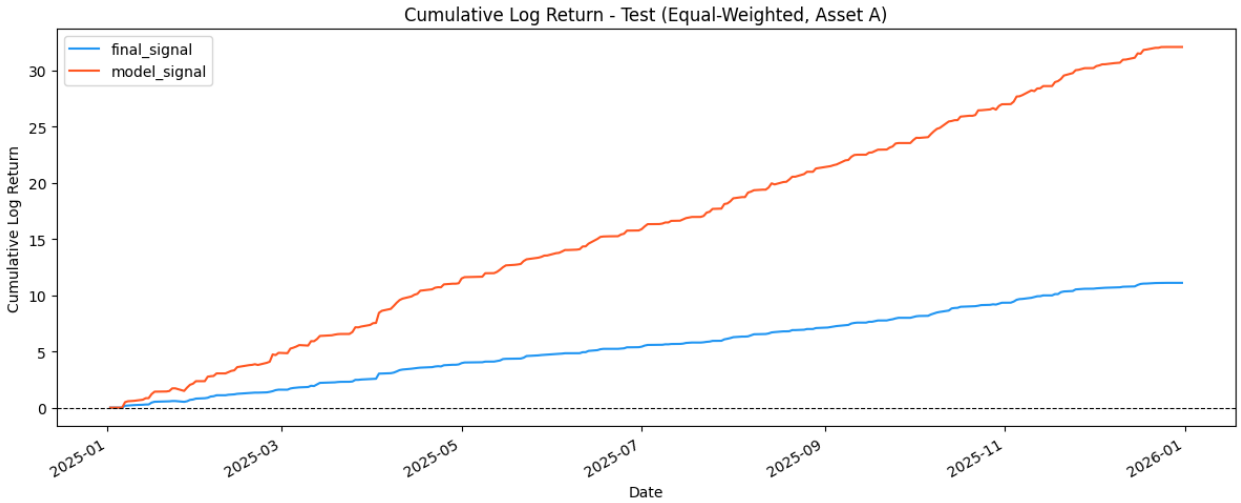

Figure 13: PnL of the testing backtest.

Finally, the backtest done on the testing period in Figures 13 and 14 yields nearly identical results to the validation backtest. This is evidence of the overall strategy being robust to pricing trend changes over at least a two-year period, suggesting that given the train/validation/test split used in this report, the out-of-sample success of the strategy will extend past a short-term period.

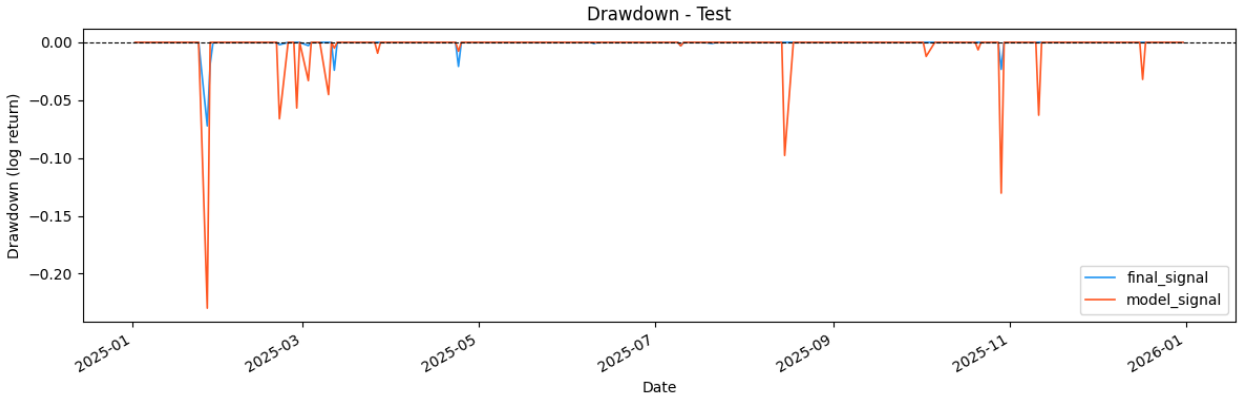

Figure 14: Drawdown plot of trades made on testing data.

Conclusion

In this report we explored pairs trading as an investment strategy and provided our own implementation on S&P 500 asset pairs using course methods. We sought to determine whether our proposed strategy could reliably detect pricing inefficiencies to signal profitable trades. By backtesting our strategy we demonstrated that, for simulated trades, the strategy is in fact profitable and may be worthwhile to implement on actual markets. Our signaling apparatus appears to be highly effective. However, we caution that the simulated PnL generated by this strategy appears to be overly optimistic and may be inflated due to a failure to consider certain factors that will affect trades on an actual market. In future investigation, we seek to implement a more rigorous backtest to ensure that it is in fact reliable and truthful.

Regardless of suspected PnL inflation, the relative performances of the model-informed trades and GMM-MinHash-informed trades introduce a fascinating topic for future exploration. In further developing this strategy, we seek to fully understand why fitting a supervised model on the produced signals produces such a significant performance enhancement. We would also like to apply additional supervised modeling techniques to see if this effect could be further exploited for maximum performance benefits.

Overall, our pair selection and signaling process was shown to corroborate the hypothesis that certain pairs of equities move together and that pricing inefficiencies between them can be detected to inform profitable trades. We observe that despite its remarkably simple thesis, pairs trading is a powerful strategy that can inspire multifaceted concrete implementations.

Finally, future development of this strategy would benefit from greater tuning of the hyperparameters at each step of our methodology, which the pipeline structure of our code enables. We would also like to acknowledge that the regimes classified by the GMMs were treated as hard regimes, which does not fully take advantage of the soft assignment capability of GMM. Future development can attempt to improve signaling effectiveness by describing mixed regimes. Once the strategy is polished on equities, a significant portion of future work could be dedicated to implementing a similar strategy in cryptocurrency markets. Given significant differences in correlations and volatility between crypto and equity markets, this venture would be fruitful to better understand what factors a successful pairs trading imple-

mentation emphasizes and how well the conceptual foundation of the strategy generalizes to different asset classes. Applying this strategy to crypto would likely require completely re-defining acceptable correlation thresholds and regime classifications. Perhaps a more robust regime classification engine could be implemented that takes into account historical data rather than being combined with a seperate historical signal. In conclusion, there are many areas of the current implementation that can be optimized for maximum PnL.

Supplemental material for this project can be found at these links:

- GitHub repository: https://github.com/spektordylan/pairs-trading-ml
- YouTube presentation: https://youtu.be/mBwwjQfDIAM

References

Avellaneda, M., & Lee, J. H. (2010). *Statistical Arbitrage in the US Equities Market*. Quantitative Finance, 10(7), 761–782. https://doi.org/10.1080/14697680903124632

Engle, R. F., & Granger, C. W. J. (1987). *Co-Integration and Error Correction: Representation, Estimation, and Testing*. Econometrica, 55(2), pp. 251–276. https://doi.org/10.2307/1913236

Fama, E. F. (1970). *Efficient Capital Markets: A Review of Theory and Empirical Work*. The Journal of Finance, 25(2), 383–417. https://doi.org/10.2307/2325486

Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). *Pairs Trading: Performance of a Relative-Value Arbitrage Rule*. The Review of Financial Studies, 19(3), pp. 797–827. http://www.jstor.org.colorado.idm.oclc.org/stable/3844014