ORIE 5741
Learning with Big Messy Data

Machine Learning Solution on ETF Pairs Trading

Linjia Feng
lf433@cornell.edu

Jingde Wan
jw2379@cornell.edu

Yitong Zou
yz2849@cornell.edu

Abstract

This project focuses on developing a pairs trading strategy using machine learning techniques to find potential trading pairs from a set of 213 Commodity-linked ETFs. We aim to use advanced data science techniques like Principal Component Analysis (PCA) and DBSCAN clustering algorithm to identify pairs of assets that have a strong correlation. Once the pairs are identified, we will use adapted linear regression approaches (rolling regression with a lookback window and Kalman filter) to help us dynamically estimate the slope and intercept (and hence hedging ratio) between a pair of ETFs in the trading model. Finally, we will backtest our model using various modules and Profit and Loss (P&L) metrics to evaluate its performance and select the better trading model. Overall, this project aims to leverage the power of machine learning to build effective pairs trading strategies that can profit from a large universe of commodity-linked ETFs.

1. Introduction

Pairs trading is a classical statistical arbitrage investment strategy that was developed in the 1980s [1]. The main idea of pairs trading is to identify two highly correlated assets and make trading signals when a statistical anomaly is detected in the spread between the two assets. This is based on the belief that the two assets' prices, which have moved closely together in the past, will continue to do so consistently in the future.

The successful pairs trading strategy relies on two things: finding the right trading pairs and designing a robust trading model to open/close market position. In this project, we aim to explore whether state-of-the-art machine learning techniques can offer valuable solutions to traditional pairs trading strategy. First, we plan to utilize unsupervised learning methods, such as PCA and clustering algorithms, to find promising clusters that contain potentially profitable pairs within the asset universe. Second, we intend to employ adapted linear regression approaches to dynamically estimate the slope and intercept, and consequently, the hedging ratio between a pair of assets in the trading model.

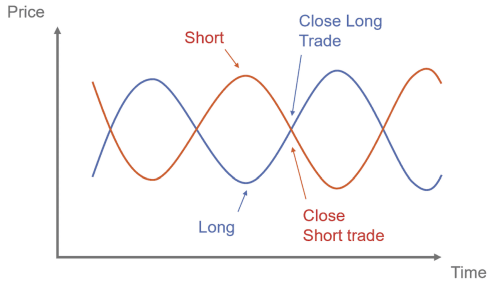

Figure 1: Pairs Trading (Image Credit: https://algotrading101.com/learn/pairs-trading-guide/)

2. Methods

2.1. PCA and Factor Model

PCA, or Principal Component Analysis, is an unsupervised learning algorithm that employs an orthogonal transformation to change a collection of potentially correlated variables into a series of linearly uncorrelated variables known as principal components. The transformation is designed so that the initial principal component captures the maximum variability within the data. Subsequent components are then determined with the highest possible variance while remaining orthogonal to the prior components [2].

According to the Arbitrage Pricing Theory (APT), securities with the same risk exposure should have the same long-run expected return. According to this assumption, a factor model for excess return can be written as[4]:

$$R_{j,t} = \beta_{0,j} + \beta_{1,j}F_{1,t} + \cdots + \beta_{p,j}F_{p,t} + \epsilon_{j,t}$$

where

- $F_{1,t}, \ldots, F_{p,t}$ are factors, which represent the "state of the financial markets and world economy" at time t
- The parameter $\beta_{i,j}$ is called a factor loading and specifies the sensitivity of the j th return to the i th factor.

According to Avellaneda and Lee (2008)[6], the process of using PCA to estimate the loadings and factors in a factor model for returns involves several steps. First, from PCA, we have

$$\mathbf{Y}_t = \sum_{i=1}^{d} \left(\mathbf{o}_i^\top \mathbf{Y}_t\right) \mathbf{o}_i$$

Then, if we use the first k PCA directions, then the j th variable has a representation as a factor model:

$$\underbrace{Y_{j,t}}_{R_{j,t}} = \sum_{i=1}^{k} \underbrace{\left(\mathbf{o}_i^\top \mathbf{Y}_t\right)}_{F_{i,t}} \underbrace{(\mathbf{o}_i)_j}_{\beta_{i,j}} + \underbrace{\sum_{i=k+1}^{d} \left(\mathbf{o}_i^\top \mathbf{Y}_t\right) (\mathbf{o}_i)_j}_{\epsilon_{j,t}}$$

which is comparable to the original factor model.

2.2. DBSCAN Clustering

DBSCAN, or Density-Based Spatial Clustering of Applications with Noise, is clustering algorithm used to group together data points that are close to each other in the feature space based on a density threshold. DBSCAN defines two important parameters, ε and $minPts$ to identify dense regions and outliers. ε is a distance measure that specifies the maximum distance between two data points for them to be considered part of the

same cluster in DBSCAN clustering. It controls the size of the neighborhood around each data point and determines the level of granularity in the clustering. $minPts$ is the minimum number of data points required to form a dense region in DBSCAN clustering. It controls the minimum size of a cluster and helps to identify clusters that are sufficiently dense [2].

We employ the DBSCAN clustering method with suitable parameter values to identify clusters of ETFs with similar characteristics based on the factors generated from PCA.

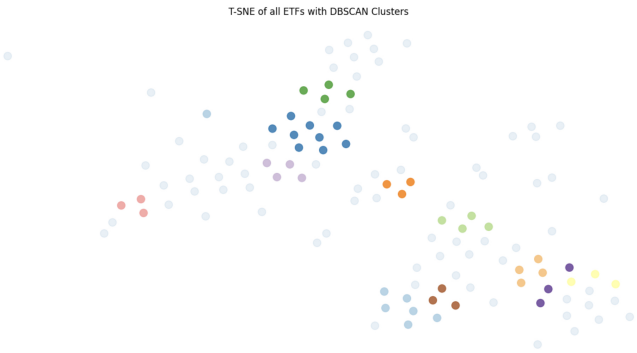

Figure 2: ETF clusters using 5 Factor Models

2.3. Conintegration Test

The concept of cointegration was initially introduced by Engle and Granger (1987). Cointegration refers to a situation where a group of variables can be combined linearly, with an order of d, resulting in a lower level of integration known as $I(d-1)$. Cointegration will be confirmed when a set of non-stationary variables I(1) can be utilized to model a stationary variable I(0). In a formal sense, if we consider two time series, y_t and x_t, both exhibiting $I(1)$ characteristics, cointegration suggests the existence of coefficients μ and β such that:

$$y_t - \beta x_t = u_t + \mu$$

where u_t is a stationary series [2].

Cointegration test are used to detect enduring and stable relationships among variable sets. The primary method for cointegration testing involves examining whether a unit root exists in the series y_t and x_t, typically through the application of the Augmented Dickey-Fuller (ADF) test. If the test confirms the presence of a unit root, a regression analysis is performed, and the resulting residuals $\hat{u}_t$ are recorded. Subsequently, the residuals are subjected to another ADF test to assess the presence of a unit root. If the null hypothesis of a unit root in the residuals is rejected, indicating that the residual series is stationary, it implies that the two variables are cointegrated [2].

2.4. Rolling Regression

The rolling regression is essentially a linear regression model performed within a continuously rolling window of data. For example, if we consider a scenario with 10 observations and a rolling window of 5 observations, we would conduct a total of 6 regression models.

In the context of simple linear regression, the approach assumes that the slope and intercept remain constant over time when calculating them.

$$Y = \beta X + \alpha$$

However, in the rolling model, we aim to incorporate the variations in slope and intercept based on the observed values. This allows us to better capture the evolving relationship between the variables and identify potential trends, patterns, or structural changes in the data.

2.5. Kalman Filter

The Kalman filter is a recursive algorithm used to estimate the state of a dynamic system with noisy measurements. In this project, the primary role of the Kalman Filter is to determine the dynamic hedge ratio between ETF pairs. According to Halls-Moore (2017)[4], multiple linear regression posits that a response value y is a linear function of its feature inputs x:

$$y(\mathbf{x}) = \beta^T \mathbf{x} + \epsilon$$

Here, $\beta^T = (\beta_0, \beta_1, \ldots, \beta_p)$ denotes the transposed vector of the intercept β_0 and slopes β_i, while $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$ signifies the error term. In this one-dimensional setting, we can simply write $\beta^T = (\beta_0, \beta_1)$ and $\mathbf{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$

We define the hidden states of our system as the vector β^T, which encompasses the intercept and slope of our linear regression. The subsequent step involves assuming that the intercept and slope for tomorrow will be equal to those of today, with the addition of some random system noise:

$$\beta_{t+1} = \mathbf{I}\beta_t + w_t$$

The transition matrix is set to the two-dimensional identity matrix, $G_t = \mathbf{I}$. This constitutes one-half of the state space model. The next step involves using one of the ETFs in the pair as the "observations".

3. Experiments and Results

3.1. Data

We choose Exchange-Traded Funds (ETFs) as the asset class to trade in this project. ETFs are a type of security that tracks a particular index, sector, commodity, or other asset and can be purchased or sold on a stock exchange in the same way that a common stock does. One advantage that ETF has is that ETF is more robust over time, which means that the past and future behaviors of the same ETF will be more consistent. The data used in this project consists of the daily close prices of a list of 213 Commodity-linked ETFs. These ETFs were selected from a larger list of ETFs that track single commodities, commodity-linked indexes or companies focused on exploring a commodity, which was obtained from ETF.com. The data collection period spans five years, from the beginning of 2015 to the end of 2019. The raw data contains the daily adjusted closing prices for each of the 213 ETFs over the five-year period. The price data for ETFs was obtained from the Tiingo API. Tiingo is a financial data provider that offers a wide range of financial data for various types of assets. The raw data exhibits a significant number of missing values that require additional processing to address.

3.2. Data Preprocessing

The data preprocessing process plays a crucial role in ensuring the quality and integrity of the dataset used for developing the pairs trading strategy. In this project, we follow a systematic approach to clean the raw data, as outlined below:

3.2.1 Imputing Missing Data

To tackle the issue of missing values in our dataset, we employ an interpolation method that assumes a linear relationship within the range of available data points. This approach, which is well-suited for time series security price data, estimates the missing values by interpolating based on known past and future data points. By assuming a linear relationship, we can capture the underlying trend and pattern in the data, thus providing a more accurate estimation of the missing values. We establish a maximum of 10 missing data points that can be filled using interpolation to maintain the dataset's reliability. This interpolation procedure ensures that the dataset is comprehensive and ready for further analysis, preserving the time series structure and minimizing potential biases that may arise from excluding incomplete observations.

3.2.2 Removing Non-Liquid ETFs

To guarantee the liquidity of the ETFs in our trading strategy, we eliminate any ETFs that did not experience trading for at least one day. This measure ensures that we focus on ETFs that possess adequate trading activity and liquidity to support our approach. After discarding the ETFs with insufficient liquidity and those that lacked active trading, we are left with 121 ETF tickers that will serve as the foundation for our subsequent analysis.

3.2.3 Splitting the Data

To evaluate the performance of our model and ensure its robustness, we split the processed data into two parts. The first three years of the dataset will serve as the training set, allowing us to find promising trading pairs. The remaining two years will be used as the test set, which provides unseen data to help us build trading models based on the trading pairs selected from the training set.

3.3. Trading Pairs

After obtaining a cleaned dataset, our next step is to identify profitable pairs for our trading strategies. Traditionally, pairs are identified by considering combinations of every security with every other security in

the dataset to check for cointegration, resulting in a total of $\frac{n\times(n-1)}{2}$ possible pairs, where n is the number of available ETFs. However, this approach presents two serious problems. Firstly, comparing every pair in the dataset is time-consuming due to the time complexity of $O(n^2)$. Secondly, this approach is susceptible to the multiple comparison problem, where each time we apply the cointegration test, there is a false positive rate of 5% due to setting the p-value to 0.05. As a result, there could be $5\% \times \frac{n\times(n-1)}{2}$ incorrectly identified cointegrated pairs. To address these issues, we propose using powerful unsupervised learning algorithms to help us identify potential trading pairs.

As mentioned above, we use Principal Component Analysis (PCA) to reduce the dimensionality of the returns data and extract historical latent common factor loadings for each ETF. We set the number of principal components to be 5, resulting in five-factor loadings for each ETF. This approach provides a concise representation of each ETF based on its time-series return data. Next, we apply an unsupervised learning algorithm, specifically clustering, to group potential ETFs based on the features produced by PCA. We chose Density-Based Spatial Clustering of Applications with Noise (DBSCAN) over K-Means due to two significant advantages: DBSCAN avoids clustering all stocks and excludes ETFs that do not fit neatly into a cluster. Ultimately, we identify 11 clusters and plot the demean time series return for 11 clusters as shown in the figures below.

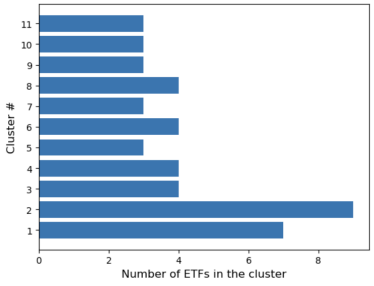

Figure 3: 11 ETF Clusters

Next, we examine whether each possible pair within a cluster is cointegrated by conducting the Engle-Granger two-step cointegration test. This involves performing an ordinary least squares regression (OLS) on the price

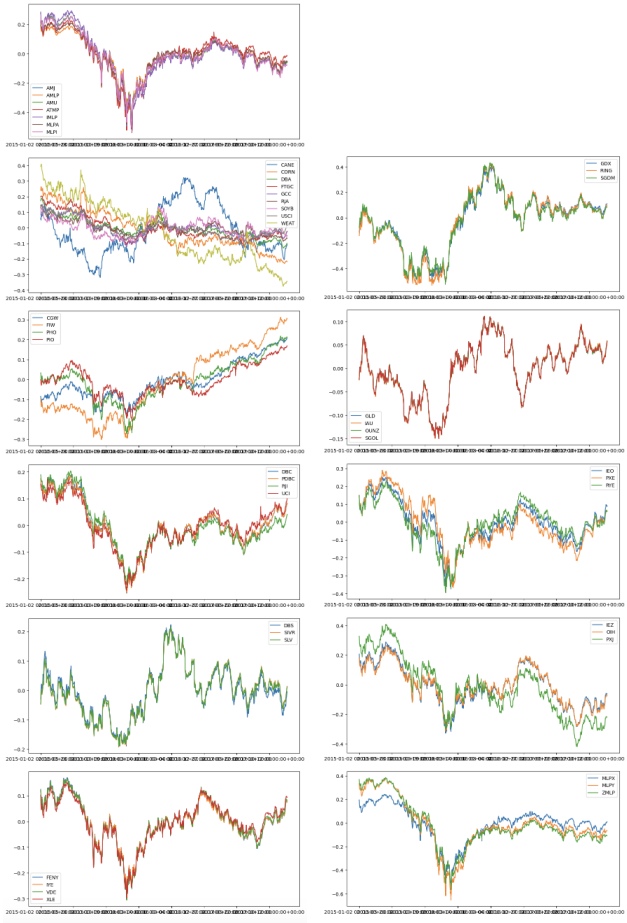

Figure 4: ETF Time Series for 11 Clusters

time series to obtain the residuals and then using the Augmented Dickey-Fuller (ADF) with $p - value$ = 0.05 test to test for unit roots in the residuals. We identify 6 promising trading pairs after conducting this analysis, as shown in the table below.

Tickers of Trading Pairs		
No.	ETF 1	ETF 2
1	AMJ	AMU
2	CORN	WEAT
3	RING	SGDM
4	GLD	OUNZ
5	GLD	SGO
6	OUNZ	SGOL

Table 1: Selected Trading Pairs

3.4. Trading Model

A significant challenge associated with this strategy is the potential time variability of the parameters introduced through the structural relationship, such as the hedge ratio between the two assets. In other words, these parameters may not remain constant throughout the strategy's duration. Identifying a method to adjust the hedge ratio over time could enhance the strategy's profitability.

One solution to this issue is to employ a rolling linear regression with a lookback window. This method entails updating the linear regression at each bar, allowing the slope and intercept terms to adapt to the most recent behavior of the cointegration relationship. An alternative approach involves using the Kalman Filter, which regards the "true" hedge ratio as a hidden, unobserved variable and seeks to estimate it using "noisy" observations.

We opted to employ linear models instead of more complex ones due to the low-frequency nature of our financial dataset (daily) and the deficiency of sample, which could lead to the issue of overfitting. Overfitting is a primary challenge that machine learning algorithms encounter when working with financial time series data [8] [9].

3.4.1 Rolling Regression-Based Trading Model

The rolling regression-based trading model, commonly referred to as Bollinger Bands, utilizes a rolling simple moving average of a spread series, along with "bands" that surround the series, which are created by multiplying the rolling standard deviation of the price series by a scalar factor. Both the moving average and standard deviation share the same lookback period. Fundamentally, these bands serve as an indicator of the current volatility in a price series.

A mean-reverting series, by definition, will occasionally deviate from its average before eventually reverting back. Bollinger Bands offer a method for entering and exiting trades by using standard deviation "thresholds" as trigger points for opening and closing positions.

Trading strategy mechanism is outlined below: First, we calculate the z-score

$$z_{score} = \frac{Spread_n - Spread_m}{\sigma_n}$$

where the subscript n, m indicate the size of look back window used in the simple moving average. Here we set $n = 30$ and $m = 1$

Second, we generate the trading signals to open/close position based on the following rules:

- If $z_{score} < -z_{open}$, we long the spread.
- If $z_{score} > -z_{close}$, we close the long position.
- If $z_{score} > z_{open}$, we short the spread.
- If $z_{score} < z_{close}$, we close the short position.

where z_{open} and z_{close} are thresholds to open and close position respectively. Here we set $z_{open} = 1$ and $z_{close} = 0.5$.

3.4.2 Rolling Regression Backtesting

We generate trading signals and record the position for each timestamp within each trading pair.Additionally, we construct an equally-weighted portfolio based on six selected trading pairs and compare its performance with relevant benchmarks. P&L metrics, such as Sharpe ratio and max drawdown, are presented below.

As evidenced by the figures and tables below, the rolling regression-based trading model fails to outperform the benchmark. In fact, its cumulative returns fall short of the 10-Year U.S. Treasury Bond, which is commonly viewed as the risk-free rate. However, the strategy's relatively low volatility and maximum drawdown suggest that the pairs trading strategy effectively mitigates risk.

3.5. Kalman Filter-Based Trading Model

Similar to the rolling regression-based trading model, the Kalman filter-based trading model also dynamically estimates the slope and intercept (and hence hedge ratio) between a pair of ETFs. Trading signals are now generated by comparing the forecast error $e_t = y_t - \hat{y}_t$,

	Portfolio	S&P 500	10 Year US Treasury
Annualized Return	0.012775	0.129221	0.024943
Annualized Volatility	0.011870	0.141725	0.000324
Sharpe Ratio	-0.608718	0.770656	nan
Max Drawdown	-0.008904	-0.197782	-0.000000

Table 2: Rolling: Equally-Weighted Portfolio P&L Metrics

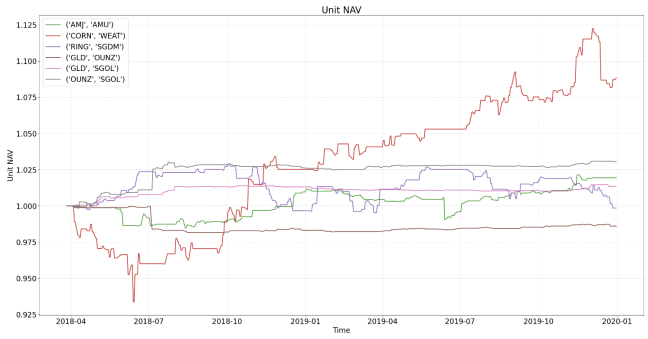

Figure 5: Rolling: Cumulative Returns for Each Trading Pair

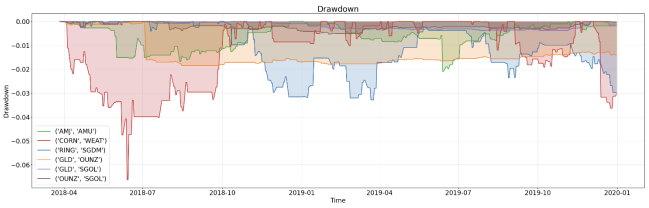

Figure 6: Rolling: Drawdowns for Each Trading Pair

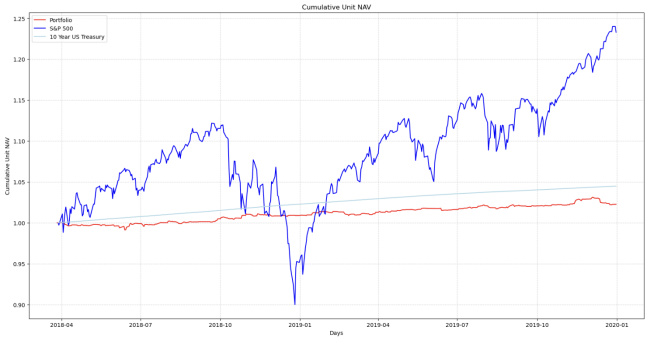

Figure 7: Rolling: Cumulative Return for Equally-Weighted Portfolio

which is the difference between y today and the estimated y of Kalman filter today and the variance of the prediction $\sqrt{Q_t}$.

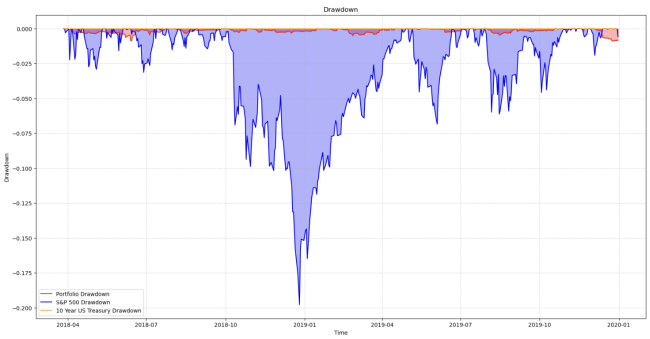

Figure 8: Rolling: Drawdowns for Equally-Weighted Portfolio

We generate the trading signals to open/close position based on the following rules [7]:

- If $e_t < -\sqrt{Q_t}$, we long 1 unit of $\mathbf{y}$ and short β_t^0 unit of $\mathbf{x}$.
- If $e_t > -k\sqrt{Q_t}$, we close the position we opened above.
- If $e_t > \sqrt{Q_t}$, we short 1 unit of $\mathbf{y}$ and long β_t^0 unit of $\mathbf{x}$.
- If $e_t < k\sqrt{Q_t}$, we close the position we opened above.

Here we set $k = 0.01$.

3.5.1 Kalman Filter Backtesting

Similarly, We generate trading signals and record the position for each timestamp within each trading pair. Additionally, we construct an equally-weighted portfolio based on six carefully selected trading pairs. We follow the same performance evaluation steps as we did in the rolling regression-based trading model.

Our observations revealed that the strategy's cumulative returns were roughly on par with those of the SP 500. However, the model demonstrated significantly lower maximum drawdown and volatility compared to the benchmark index.

This suggests that the Kalman filter-based trading model can potentially offer investors a more stable return profile, with reduced risk in terms of drawdowns and fluctuations, while still generating competitive returns. The

strategy's ability to control risk may be particularly appealing to investors seeking exposure to the market while mitigating downside potential.

	Portfolio	S&P 500	10 Year US Treasury
Annualized Return	0.050134	0.100809	0.025256
Annualized Volatility	0.013618	0.149546	0.000311
Sharpe Ratio	2.212841	0.540360	nan
Max Drawdown	-0.006020	-0.197782	-0.000000

Table 3: KF: Equally-Weighted Portfolio P&L Metrics

Figure 9: KF: Cumulative Returns for Each Trading Pair

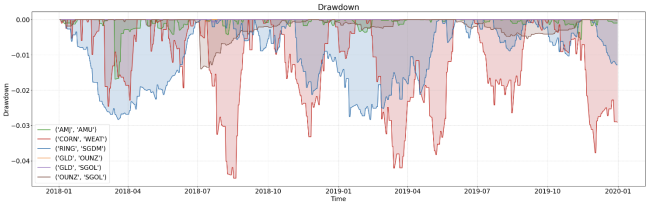

Figure 10: KF: Drawdowns for Each Trading Pair

4. Fairness

Fairness is an essential criterion when choosing a model for any application, as it ensures equitable treatment of all parties involved. For our project, fairness is reflected in market neutrality: our model is designed to be market-neutral, meaning it doesn't favor one ETF over the other. This ensures equal opportunity for profit-making and minimizes any potential bias.

5. Contributions

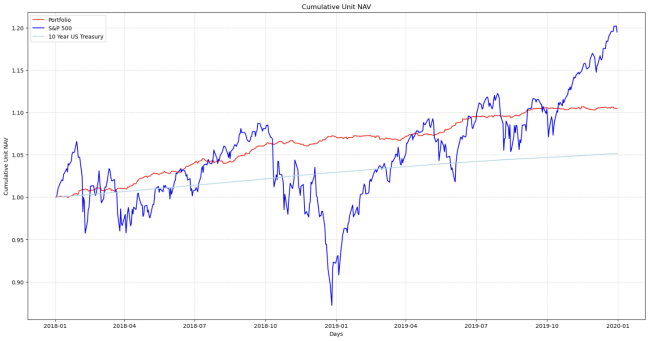

Figure 11: KF: Cumulative Return for Equal-Weighted Portfolio

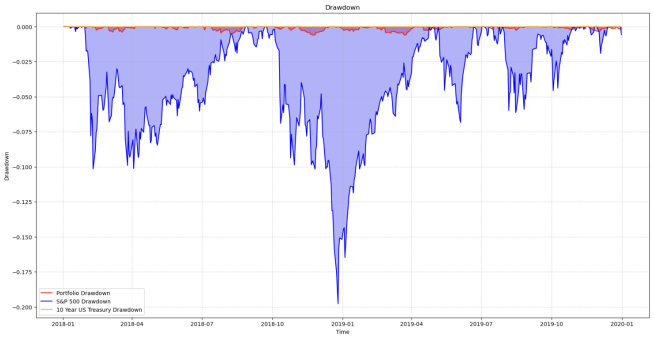

Figure 12: KF: Drawdowns for Equal-Weighted Portfolio

- Linjia Feng: rolling regression, report, slide
- Jingde Wan: data preprocessing, PCA, clustering, rolling regression, Kalman filter, report
- Yitong Zou: rolling regression, report, slide

6. Conclusion

In conclusion, our experiments have demonstrated that unsupervised learning techniques, including PCA and DBSCAN clustering, effectively help us identify profitable trading pairs while substantially reducing time complexity compared to traditional methods. Additionally, adpated linear regression methods, such as rolling regression and the Kalman filter, enable us to construct trading models that dynamically estimate hedge ratios, effectively mitigating market risk. Notably, the Kalman filter-based trading model yields a considerable amount of P&L, suggesting its potential for application in a production environment.

References

[1] Gatev, E., Goetzmann, W. N., and Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative-value arbitrage rule. Review of Financial Studies, 19(3):797–827

[2] Sarmento, S. M., Horta, N. (2021). A Machine Learning Based Pairs Trading Investment Strategy. Springer International Publishing.

[3] Engle, R.F. and Granger, C.W. (1987). Co-integration and error correction: representation, estimation, and testing. Econ: J Econo Soc 251–276.

[4] Halls-Moore, M. (2017). Advanced Algorithmic Trading. QuantStart.

[5] Tsay, R. S. (2005). Analysis of financial time series (2nd ed.). John Wiley Sons.

[6] Avellaneda, M., Lee, J. H. (2010). Statistical arbitrage in the US equities market. Quantitative Finance, 10(7), 761-782.

[7] Chan, E. P. (2013). Algorithmic trading: Winning strategies and their rationale. John Wiley Sons.

[8] De Prado, M. L. (2018). Advances in financial machine learning. John Wiley Sons.

[9] Chan, E. P. (2017). Machine trading: Deploying computer algorithms to conquer the markets. John Wiley Sons.