Clustering-based pair trading using Reinforcement learning

Leung Pak Hei, Marco

Paper ID

Abstract

This project aims to compare clustering-based and reinforcement learning-based pair trading with traditional pair trading strategies, in the context of stock trading. The project is implemented using Python programming language and Interactive Brokers API for accessing real-time market data and trading functionality.
Reinforcement learning algorithms are implemented to learn and adapt from experience in the stock market, while traditional trading strategies are implemented to identify and take advantage of market inefficiencies. The performance of these approaches is compared based on various metrics, including return on investment, risk-adjusted return, and Sharpe ratio.
The results of the project show that clustering and reinforcement learning-based pair trading outperform traditional trading strategies in terms of both return on investment and Satino ratio. The analysis of the strengths and weaknesses of these approaches provides valuable insights into the effectiveness of reinforcement learning and clustering in the stock market.
Overall, this project explores different reinforcement learning algorithms, and clustering algorithms, and demonstrates the potential of reinforcement learning-based pair trading.

1. Introduction

Reinforcement learning has shown great potential in various fields, including finance and trading. Stock trading is one of the areas where reinforcement learning has gained significant attention in recent years. The ability to learn and adapt from experience makes reinforcement learning a promising approach for stock trading, where the market dynamics and conditions are constantly changing.
The goal of this project is to compare reinforcement learning-based trading with traditional trading strategies, including pairs trading, momentum trading, and portfolio optimization. Pairs trading involves identifying two stocks that have a high correlation and taking advantage of any temporary price divergences between the two stocks. Python is a popular language for implementing reinforcement learning algorithms, and Interactive Brokers API provides access to real-time market data and trading functionality. In this project, we will use Python and Interactive Brokers API to implement and compare the performance of reinforcement learning-based trading and traditional trading strategies.
The main objectives of this project are:
1. Implementing reinforcement learning algorithms for stock trading.
2. Compare different types of clustering methods.
3. Compare different reinforcement learning algorithms.
4. Fune-tinning parameters to find optimal trading pairs.
5. Back-tested the performance of reinforcement learning-based pair trading.
The comparison of reinforcement learning-based trading with traditional trading strategies can provide valuable insights into the effectiveness of these approaches in the stock market. The results of this project can be useful for investors and traders who are looking for new and effective trading strategies.

2. Methodology

2.1. Problem description

The problem addressed in this project is to compare reinforcement learning-based pair trading with traditional pairs trading. The objective is to evaluate the performance of these approaches based on various metrics, including return on investment, risk-adjusted return, and Sharpe ratio.

2.2. Dataset

The dataset used in this project consists of historical stock prices and market data, which is obtained using the Interactive Brokers API. The objective is primarily based on the stocks under S&P500 and the training dataset consists of stock data from 2017 to 2020, and the testing data set consists of stock data from 2021 to Mar 2023. For simplicity, this project focuses only on the ”day” interval bar chart data.

3. Implementation

3.1. Clustering Algorithms

In this project, different clustering algorithms are implemented to classify the stocks under S & P 500 into different groups which include, K-means clustering, Hierarchical Clustering, Affinity Propagation, DBSCAN, Gaussian Mixture Model, and Ordering Points to Identify the Clustering Structure(OPTICS). The clustering will be primarily based on the historical standard deviation and the expected return in the training period.

3.1.1 K-means

K-means clustering is a popular unsupervised machine learning algorithm that can be used to group similar data points together. In your scenario, K-means clustering can be used to group stocks under SP 500 based on their historical standard deviation and expected return in the training period. Let X be an n x d matrix where n is the number of stocks and d is the number of features (in this case, 2: historical standard deviation and expected return). The algorithm aims to partition the data points into K clusters such that the sum of squared distances between each data point and its assigned centroid is minimized. The algorithm starts by randomly initializing K centroids and then iteratively assigns each data point to the nearest centroid, updates the centroids based on the new assignments, and repeats the process until convergence. The objective function for K-means clustering can be defined as $J = \sum_{i=1}^{K} \sum_{x_j \in S_i} ||x_j - c_i||^2$, where S_i is the set of data points assigned to the ith centroid. The algorithm aims to find the values of the centroids that minimize J.

3.1.2 Hierarchical Clustering

Hierarchical clustering is a clustering algorithm that creates a hierarchy of clusters. There are two types of hierarchical clustering: agglomerative and divisive. Agglomerative clustering starts with each data point as a separate cluster and then iteratively merges the closest pairs of clusters until a single cluster is formed. Divisive clustering starts with all data points in a single cluster and then recursively splits the cluster into smaller clusters until each cluster contains a single data point. In this response, we will focus on agglomerative hierarchical clustering.

Agglomerative hierarchical clustering starts by assigning each data point to a separate cluster. The algorithm then iteratively merges the closest pairs of clusters until a single cluster is formed. The distance between two clusters is defined in terms of the distance between their constituent data points. There are several methods for computing the distance between clusters, including single linkage, complete linkage, average linkage, and Ward's method.

Single linkage computes the distance between two clusters as the minimum distance between any pair of data points in the two clusters. Complete linkage computes the distance between two clusters as the maximum distance between any pair of data points in the two clusters. Average linkage computes the distance between two clusters as the average distance between all pairs of data points in the two clusters. Ward's method computes the distance between two clusters as the increase in the sum of squares of the distances between the data points in the two clusters when the clusters are merged.

The merging of two clusters is represented by a binary tree called a dendrogram. The dendrogram shows the hierarchy of clusters and the distances between them. The leaves of the dendrogram represent the data points, and the internal nodes represent the merged clusters. The height of each internal node corresponds to the distance between the merged clusters. The dendrogram can be cut at a certain height to obtain a partition of the data into a specified number of clusters.

The agglomerative hierarchical clustering algorithm can be summarized as follows:

1. Assign each data point to a separate cluster.
2. Compute the distance between all pairs of clusters.
3. Merge the closest pair of clusters.
4. Recompute the distance between the merged cluster and all other clusters.
5. Repeat steps 3-4 until all data points are in a single cluster.

The resulting dendrogram can be cut at a certain height to obtain a partition of the data into a specified number of clusters. The choice of the height at which to cut the dendrogram depends on the specific problem and can be determined using various methods, including the elbow method and the silhouette method. [5]

3.1.3 DBSCAN

The DBSCAN algorithm is based on the concept of density reachability. A data point x is considered to be density reachable from another point y if there exists a chain of data points $x1, x2, ..., x_n$ such that $x_1 = yandx_n = x$, and for all i, $1in, xi + 1$ is within the epsilon neighborhood of xi. The epsilon neighborhood of a data point x is defined as the set of all data points within a distance of epsilon from x. [5]

A data point is considered a core point if it has at least minPts data points within its epsilon neighborhood. A border point is a data point that is not a core point but is in the epsilon neighborhood of a core point. A noise point is a data point that is not a core point and is not in the epsilon neighborhood of any core point.

The DBSCAN algorithm starts by randomly selecting a

data point and finding all its neighboring data points within the epsilon neighborhood. If the number of neighboring data points is greater than or equal to minPts, the data point is labeled as a core point. The algorithm then expands the cluster by adding all the data points that are within the epsilon neighborhood of any core point. The algorithm continues until all data points are assigned to a cluster.
The DBSCAN algorithm can be summarized as follows:
1. Randomly select a data point x.
2. Find all data points within the epsilon neighborhood of x.
3. If x has at least minPts data points within its epsilon neighborhood, label x as a core point and add it to the current cluster.
4. Expand the current cluster by adding all data points that are within the epsilon neighborhood of any core point.
Repeat steps 1-4 until all data points are assigned to a cluster.

3.1.4 Affinity Propagation

Affinity Propagation is a clustering algorithm that does not require the number of clusters to be specified in advance. Instead, it uses message passing between data points to determine the number of clusters and their exemplars. In your scenario, Affinity Propagation can be used to group stocks under SP 500 based on their historical standard deviation and expected return in the training period. Let X be an n x d matrix where n is the number of stocks and d is the number of features (in this case, 2: historical standard deviation and expected return).
The Affinity Propagation algorithm is based on the concept of "responsibility" and "availability" between data points. The responsibility matrix R and availability matrix A are used to update the exemplars and clusters. The algorithm starts by initializing the matrices R and A and iteratively updates them until convergence. The exemplars and clusters can be identified from the resulting matrices. [2]
The responsibility between two data points i and j is defined as the suitability of data point j to be the exemplar of data point i. The availability of data point j is defined as the suitability of data point i to choose data point j as its exemplar. The responsibility and availability matrices are updated iteratively using the following equations:
$r_{i,j} = s_{i,j} - max_{k \neq j}(a_{i,k} + s_{i,k})$
$a_{i,j} = min(0, r_{j,j} + \sum_{k \neq i,j} max(0, r_{k,j}))$
where $s_{i,j}$ is the similarity between data points i and j, and $r_{i,j}$ and $a_{i,j}$ are the entries in the responsibility and availability matrices, respectively.
The exemplars are the data points that have the highest sum of their responsibility and availability values. The clusters are formed by assigning each data point to the exemplar with the highest responsibility value.
The Affinity Propagation algorithm is sensitive to the choice of the similarity measure used to compute the similarity between data points. The similarity measure can be chosen based on the specific characteristics of the data.

3.1.5 Gaussian Mixture Model

Gaussian Mixture Model (GMM) is a probabilistic clustering algorithm that models the distribution of the data using a mixture of Gaussian distributions. In your scenario, GMM can be used to group stocks under SP 500 based on their historical standard deviation and expected return in the training period. Let X be an n x d matrix where n is the number of stocks and d is the number of features (in this case, 2: historical standard deviation and expected return). [7]
The GMM algorithm assumes that the data points are generated from a mixture of K Gaussian distributions, each with its own mean and covariance matrix. The probability density function of the mixture model is given by:
$P(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \sum_k)$
where π_k is the mixing coefficient for the kth Gaussian distribution, $N(x|\mu_k, \sum_k)$ is the Gaussian probability density function with mean μ_k and covariance matrix $\sum_k$, and $\sum_{k=1}^{K} \pi_k = 1$.
To estimate the parameters of the GMM, the algorithm starts by randomly initializing the parameters of K Gaussian distributions and then iteratively updates the parameters to maximize the likelihood of the data. The algorithm assigns each data point to the cluster with the highest probability of generating that data point.
The parameters of the GMM can be estimated using the Expectation-Maximization (EM) algorithm. The EM algorithm consists of two steps: the E-step and the M-step. In the E-step, the algorithm computes the posterior probabilities of each data point belonging to each cluster, given the current estimate of the parameters. In the M-step, the algorithm updates the parameters of each Gaussian distribution based on the posterior probabilities computed in the E-step. The algorithm iterates between the E-step and the M-step until convergence.

3.1.6 OPTICS

The OPTICS algorithm is based on the concept of reachability distance, which is a more flexible measure of density than the epsilon neighborhood used in DBSCAN. The reachability distance between two data points x and y is defined as the maximum distance between any two consecutive data points in a density-based cluster, starting from x and ending at y. [3]
The OPTICS algorithm constructs a reachability graph based on the distance between data points and a distance threshold called the epsilon parameter. The reachability

graph is a directed graph where each data point is a node and the edges represent the reachability distances between pairs of data points. The algorithm then identifies the density-based clusters by extracting the local maxima from the reachability plot.
The OPTICS algorithm can be summarized as follows:
1. Compute the pairwise distances between all data points.
2. For each data point x, compute its reachability distance to all other data points.
3. Construct a reachability graph where each data point is a node, and the edges represent the reachability distances between pairs of data points.
4. Find the local maxima in the reachability plot to identify the density-based clusters.
5. Assign each data point to a cluster based on its reachability distance to the exemplar of the cluster.
The OPTICS algorithm is more flexible than DBSCAN in terms of the shape and size of the clusters. It can handle clusters of different shapes and sizes and can provide a hierarchical clustering structure by using the reachability distances between data points. However, it can be computationally expensive, especially for large datasets, and requires the tuning of several parameters, including epsilon and minPts.

3.2. Reinforcement Learning Algorithms

In this project, three different reinforcement learning models are explored: DDPG, Q-Learning and deep Q-Learning, as well as different policy method and optimization algorithms.

3.2.1 Q-Learning

Q-Learning is a model-free reinforcement learning algorithm that learns by iteratively estimating the action-value function, which represents the expected reward of taking an action in a given state. In this project, Q-Learning is used to learn the optimal trading policy for a given stock.
The Q-Learning algorithm involves the following steps:
1. Initialize the action-value function for all state-action pairs.
2. Observe the current state of the stock market.
3. Choose an action based on the current state using an exploration-exploitation strategy (e.g., epsilon-greedy).
4. Execute the chosen action and observe the reward and the next state.
5. Update the action-value function for the current state-action pair using the Bellman equation.
6. Repeat the above steps until convergence.
The Bellman equation is given by: $Q(s,a) = Q(s,a) + (r + max'_a(Q(s',a')) - Q(s,a))$
where $Q(s,a)$ is the action-value function for state s and action a, r is the reward, s' is the next state, a' is the next action, is the learning rate, and is the discount factor.

3.2.2 Deep Q-Learning

Deep Q-Learning is an extension of Q-Learning that uses a deep neural network to approximate the action-value function. In this project, Deep Q-Learning is used to learn the optimal trading policy for a given stock.
The Deep Q-Learning algorithm involves the following steps:
1. Initialize a deep neural network with random weights to approximate the action-value function.
2. Observe the current state of the stock market.
3. Choose an action based on the current state using an exploration-exploitation strategy (e.g., epsilon-greedy).
4. Execute the chosen action and observe the reward and the next state.
5. Store the experience tuple (s, a, r, s') in a replay buffer.
6. Sample a batch of experience tuples from the replay buffer.
7. Compute the target Q-values using the Bellman equation and the deep neural network.
8. Update the weights of the deep neural network using gradient descent to minimize the mean squared error between the predicted Q-values and the target Q-values.
9. Repeat the above steps until convergence.
The target Q-values are given by:
$y = r + max'_a(Q(s', a',_i))$ where y is the target Q-value, r is the reward, s' is the next state, a' is the next action, γ is the discount factor, and $_i$ are the weights of the deep neural network at iteration i. [6]

3.3. Policy update

Policy optimization/update is an essential component of reinforcement learning algorithms that involves updating the policy based on the observed rewards and states. In the context of stock trading, the policy is the trading strategy that the algorithm uses to make decisions about when to buy, sell or hold a particular stock or portfolio.
There are several approaches to policy optimization/update in reinforcement learning models, including Q-learning, SARSA, and Actor-Critic methods. The Q-learning algorithm is a value-based approach that updates the Q-values of the state-action pairs based on the observed rewards and states. The SARSA algorithm is another value-based approach that updates the Q-values based on the observed state-action-reward-state-next action pairs. The Actor-Critic algorithm is a model-based approach that combines the value function approximation and the policy gradient methods to update the policy and the value function.
In the context of stock trading, the policy can be updated by

retraining the reinforcement learning model with new data, or by fine-tuning the existing model with additional training. This process is typically done in batches, with new data being added periodically to improve the accuracy of the model.

The policy optimization/update process can also involve techniques such as exploration-exploitation trade-offs, which balance the exploration of new strategies with the exploitation of known profitable strategies. This is important because in stock trading, the optimal trading policy can change over time due to changes in the market conditions and other external factors.

Overall, policy optimization/update is a critical component of reinforcement learning models in stock trading, as it allows the algorithm to adapt to changing market conditions and learn optimal trading strategies over time.

3.4. Trading strategies

3.4.1 Pairs Trading

Pairs trading is a statistical arbitrage strategy that involves identifying pairs of stocks that are co-integrated, meaning they have a long-term relationship that is not affected by short-term fluctuations in the stock market. The basic idea behind pairs trading is to go long on the underperforming stock and short on the overperforming stock in the pair, with the expectation that the underperforming stock will eventually rise and the overperforming stock will eventually fall, resulting in a profit.

Reinforcement learning models can be used to automate the process of identifying co-integrated pairs and executing trades based on the identified pairs. Specifically, a Q-Learning or Deep Q-Learning algorithm can be trained to learn the optimal trading policy for a given pair of stocks based on historical data. The state of the stock market can be defined as a set of features that capture the current market conditions, such as the stock prices, trading volumes, and technical indicators. The action space can be defined as a set of possible trades, such as going long on the underperforming stock, going short on the overperforming stock, or doing nothing. The reward function can be defined as the profit or loss resulting from the trade.

Once the Q-Learning or Deep Q-Learning algorithm is trained, it can be used to identify co-integrated pairs in real-time and execute trades based on the learned policy. This can be done by feeding the current state of the stock market into the algorithm and selecting the action with the highest Q-value.

4. Results

4.1. Comparison on the Clustering algorithms

After the clustering algorithms, different optimization is done on the parameters of the statistical value used in all of the clusterings. Then, they are compared based on the number of clusters, trading pairs, and silhouette score.

4.1.1 Comparison on the clustering image

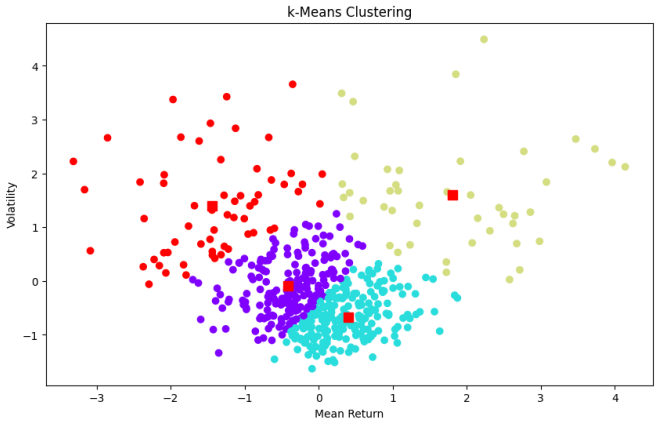

Figure 1. K-means

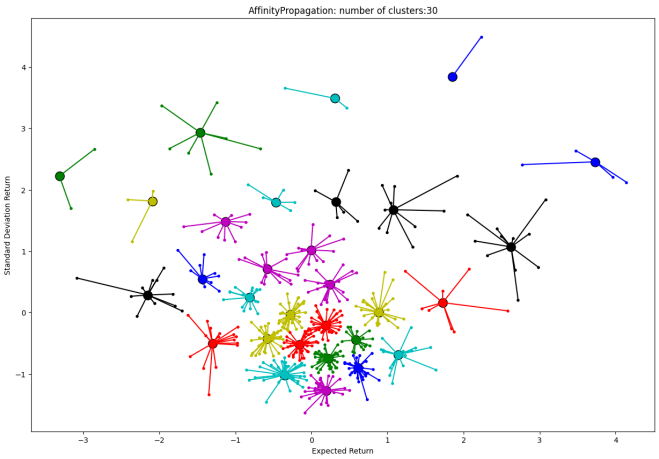

Figure 2. Affinity Propagation

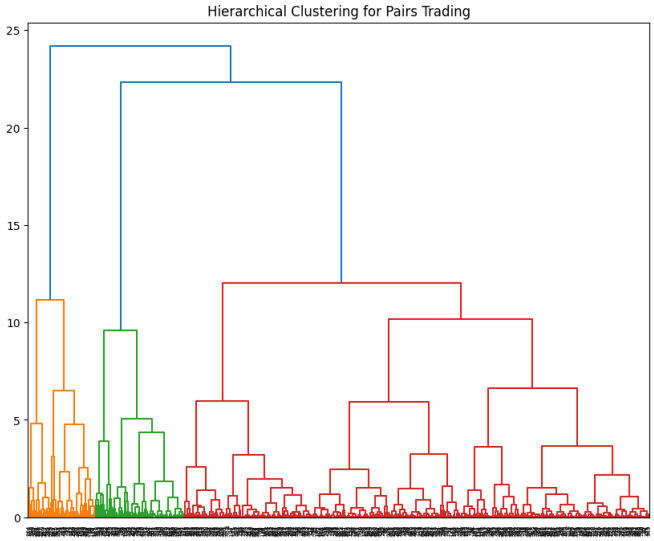

Figure 3. Hierarchical Clustering

Figure 5. OPTICS

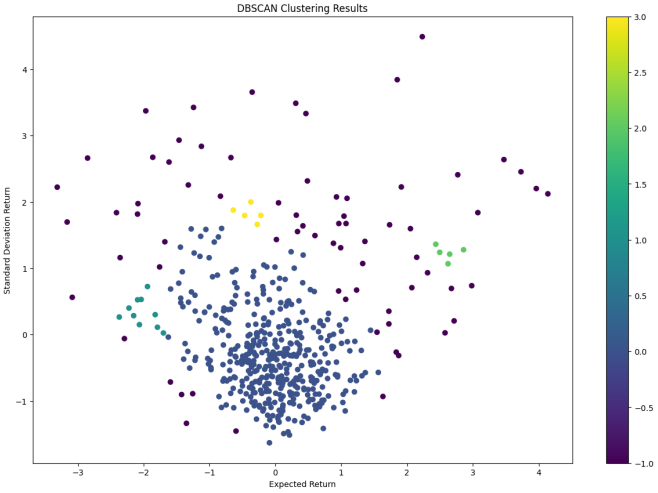

Figure 4. DBSCAN

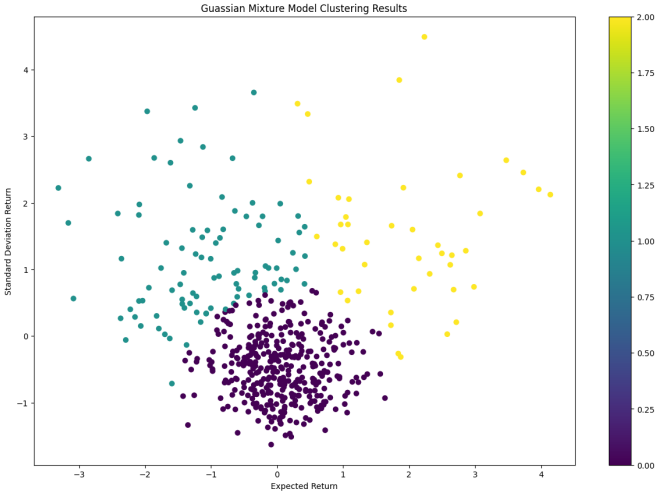

Figure 6. Gaussian Mixture Model

4.1.2 Comparison on the Number of clusters

Comparison of number of clusters	
Clustering method	Number of clusters
K-means	5
Affinity Propagation	30
Hierarchical	30
DBSCAN (eps=0.5, min sample=4)	3
OPTICS (min sample=4)	153
Gaussian Mixture Model	3

4.1.3 Comparison on the Number of silhouette score

Comparison of silhouette score	
Clustering method	Number of clusters
K-means	0.3179
Affinity Propagation	0.34607
Simple Hierarchical (top-down)	0.2775
DBSCAN (eps=0.5, min sample=4)	0.47661
OPTICS (min sample=4)	0.3371
Gaussian Mixture Model	0.48190

4.1.4 Evaluation on the Clustering algorithms

We evaluated several clustering algorithms on our dataset using the silhouette score as a performance metric. The silhouette score measures the quality of clustering by computing the average distance between each data point and its own cluster compared to the nearest neighboring cluster. A higher silhouette score indicates better clustering performance.

Based on our results, DBSCAN and Gaussian Mixture Model (GMM) performed the best, achieving the highest silhouette scores. Additionally, it is found that they are relatively fast and easy to implement.

K-means and Affinity Propagation performed the second best, achieving good silhouette scores.

OPTICS and Hierarchical clustering performed the worst, achieving the lowest silhouette scores. Additionally, it is found that it is sensitive to outliers and is very computationally expensive for large datasets.

Due to the above results and simplicity, in this project, Gaussian Mixture model clustering would be adopted and the pair trading would be primarily based on the clustering results of Gaussian Mixture model clustering.

4.2. Pairs selection

After the clustering using K-means, it is found that there are 30906 pairs are found. Note that not all of them are promising. Therefore, different decision rules are used to get the most promising trading pairs. It was first calculated the correlation matrix between all possible pairs. Based on this, we applied Gaussian Mixture model clustering using and Euclidean distance to group the pairs into 4 clusters. Within each cluster, further filtered the pairs based on statistical cointegration tests like the Augmented Dickey-Fuller test and Johansen test. Pairs that exhibited a stable cointegrating vector according to these tests were selected. Such tests are carried out in the training period with optimization of the parameters or the threshold value we use in the selection of the pair. The following is the threshold value for the pair's selection and the number of pairs left after the selection. [1] [4]

4.2.1 Correlation

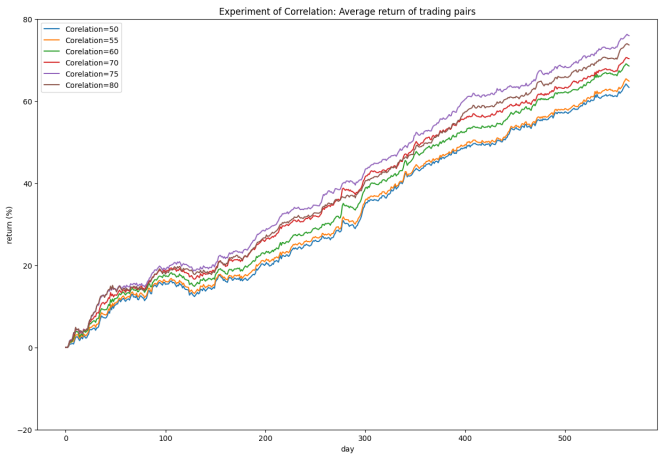

Figure 7. Correlation test

4.2.2 Cointegration

4.2.3 Standard Deviation

4.2.4 Augmented Dickey-Fuller (ADF) mean reversion test

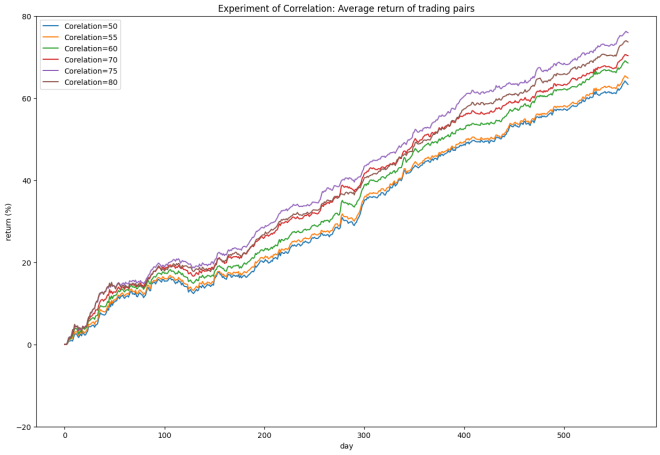

Figure 8. Cointegration Test

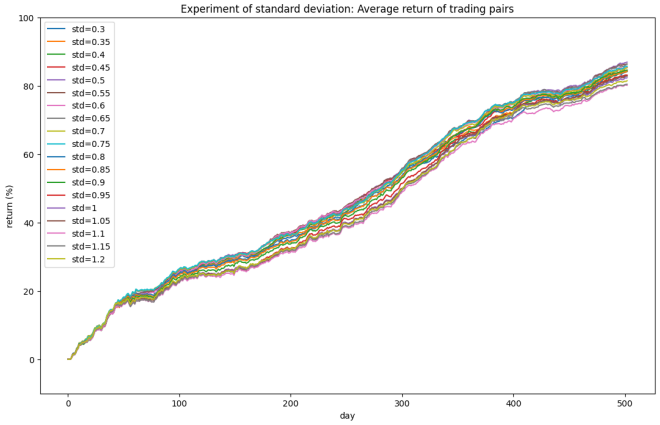

Figure 9. Volatility test

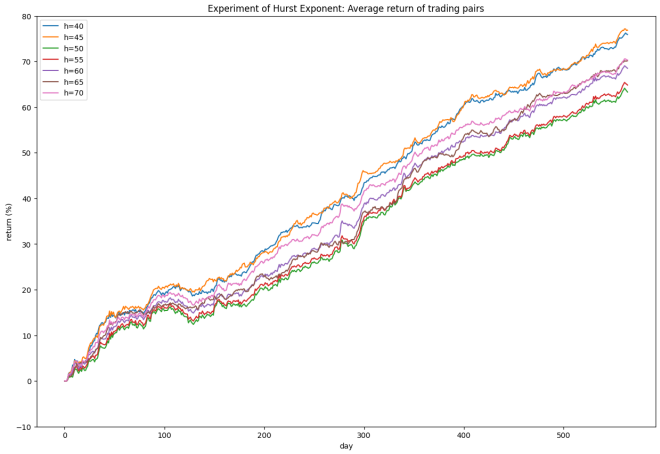

Figure 10. Mean-reversion behavior test on hurst expoenent

4.2.5 Evaluation of the trading pairs selection

66 trading pairs were identified that passed our clustering and cointegration criteria. These pairs exhibit stable mean-reverting behavior that can be exploited using a pair trading strategy. As a next step, reinforcement learning algorithms were applied to learn an optimal trading policy for these 66 pairs by modeling the cointegration vectors and volatility of each pair.

4.3. Trading implementations using reinforcement learning

In this project, the reward defined is the profit/loss from the previous. The action space defined is -1,0,1 denoted as sell, buy and hold. For the environment variables, it consists of different parameters, such as the spread of the pairs, normalized trading pairs, market volatility, Bi-LSTM predicted price, and technical indicators (Moving averages, Moving averages convergence and divergence, relative strength index, etc). Different policy gradient and reinforcement learning algorithms are used to compare and get the optimal models. The reinforcement learning model will determine when to open or close a pair trade to maximize profits while the pairs temporarily deviate from their equilibrium. By modeling multiple pairs together based on their cluster relationship, the model can also determine how to allocate capital across pairs to maximize the overall risk-adjusted returns.

4.3.1 Comparison of policy mechanisms

For the comparison of policy mechanism, epsilon-greedy exploration, entropy, and shape ratio are used.

Policy comparison		
Method	Epsilon-Greedy Exploration	Entropy
Deterministic Policy Gradient (DPG)	0.45	1.23
Deep Deterministic Policy Gradient (DDPG)	0.41	1.45
Proximal Policy Optimization (PPO)	0.32	1.7
Trust Region Policy Optimization (TRPO)	0.56	1.09
Actor-Critic	0.39	1.34
Q-Learning	0.38	1.37
SARSA	0.61	0.96

5. Back-testing Performance

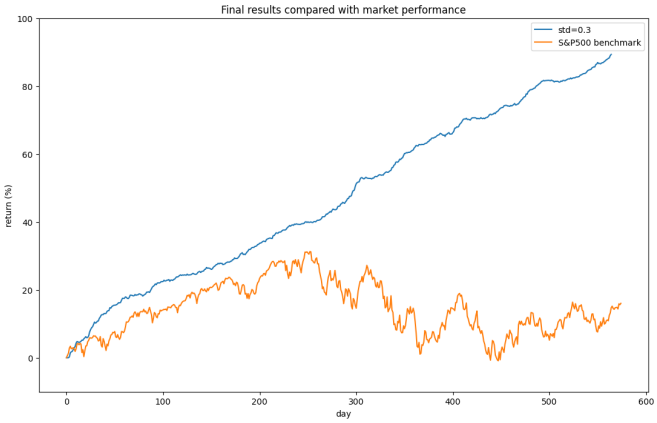

Figure 11. Back-testing results in testing period (2021 to 2022

From the above chart, it is shown that there is a stable uptrend, with little fluctuation. It outperformed the market performance and reached a sharpe ratio of 1.87, which can be classified as a satisfactory performance.

6. Discussion

6.1. Comparison of our model with related journals

There exist different journals about pair trading. However, most of them are primarily based on the statistical test or test inside the same industry, having an assumption that stocks inside the industries should have similar price movement, due to statistical arbitrage trading opportunities exist. Our research is primarily based on the global stocks under SP500 and utilizes machine learning models. Here is the comparison of our trading performance compared to other related traditional journals.

Pair Trading comparison					
Name	Time period	Asset	Technique	No. pairs	Sharpe ratio
RL with GM (mine)	2021-2023	SP500	Clustering, correlation, cointegration, mean-reversion, RL	66	1.87
Gatev (2006)	1962-2002	US stocks	spread	29	0.41
Avellaneda (2010)	1990-2008	SP100	cointegration, mean reversion	60	0.7
Cakici (2011)	1990-2008	SP500	cointegration, mean reversion	35	0.42
Bonomo (2013)	2002-2011	Bovespa index	cointegration	11	0.46

Although the asset used, and time period are different this may deviate from the fairness of the experiments. But in general, our strategy outperformed most of the experiments done by other journals using traditional approaches.

6.2. Potential improvements

6.2.1 Feature Engineering

In this project, the environment variables space is small The performance of the trading strategy can be improved by including more relevant features that capture additional information about the market dynamics. For example, incorporating sentiment analysis of news articles or social media posts could provide valuable insights into the market sentiment and help identify profitable trading opportunities.

6.2.2 Hyper-parameter tuning

In this project, hyperparameters are not tuned precisely, such as the threshold used in the trading Paris selection and reinforcement learning. The performance of the clustering and reinforcement learning algorithms can be further optimized by tuning the hyperparameters. This involves systematically testing different combinations of hyperparameters to find the optimal values that maximize the performance of the trading strategy.

6.2.3 Risk Management

The performance of the trading strategy can be further enhanced by incorporating risk management techniques, such as stop-loss orders or position sizing limits. These techniques can help control the downside risk and prevent large losses during market downturns.

7. Conclusion

Based on the results of this project, it can be concluded that clustering-based pair trading using reinforcement learning is a promising approach for developing effective trading strategies. The project tested different clustering methods and reinforcement learning algorithms and achieved a Sharpe ratio of 1.87, which outperformed the market performance. This suggests that the combination of clustering and reinforcement learning can help identify profitable trading opportunities and exploit them effectively. The success of this project highlights the potential of using machine learning techniques to enhance the performance of financial trading strategies. Future work could include further refining the clustering and reinforcement learning algorithms, testing the approach on different markets and asset classes, and exploring the use of additional data sources and features to improve the performance of the trading strategy. Overall, this project demonstrates the value of using machine learning to develop innovative and effective trading strategies that can generate superior returns in the financial markets.

References

[1] A new pairs trading algorithm using mean reversion and mean crossing strategies. 19(3):137–143, 1996. 7

[2] Li J Chen, X. Pairs trading strategy based on deep reinforcement learning and clustering. *Journal of Intelligent Fuzzy Systems*, 37(1):1273–1282, 2019. 3

[3] Kriegel H. P. Sander J. Xu X Ester, M. A density-based algorithm for discovering clusters in large spatial databases with noise. *In KDD-96 Proceedings*, 96(34):226–231, 1996. 3

[4] Chen Y. Zhang L Guo, Y. A novel pairs trading strategy based on mean reversion and mean crossing. 37(2):1513–1522, 2019. 7

[5] Kim Y. Hong, H. Clustering-based pairs trading using deep reinforcement learning. expert systems with applications, 96, 43-54, 2018. 2

[6] Taylor S. D. Ritter, G. Machine learning for trading. in handbook of systemic risk. *Cambridge University Press*, pages 555–576, 2017. 4

[7] Zhang W Zhang, Y. Clustering-based deep reinforcement learning for pairs trading. *IEEE Transactions on Neural Networks and Learning Systems*, 32(3):1052–1063, 2021. 3