A survey of statistical arbitrage pair trading with machine learning, deep learning, and reinforcement learning methods

Sun Yufei

Department of Quantitative Finance, Faculty of Economics, University of Warsaw

Abstract

Pair trading stands as a foundational strategy in quantitative trading, consistently garnering attention from economists and computer scientists alike. Over recent decades, the focus of research has broadened from linear models to encompass a range of innovative approaches, including machine learning (ML), deep learning (DL), reinforcement learning (RL), and deep reinforcement learning (DRL). These advanced models have demonstrated their capability to discern complex patterns in financial market data, thus enhancing predictive accuracy.

Increasingly, investors are leveraging the capabilities of deep learning to predict and analyze movements within the stock and foreign exchange markets, exploiting the strategic edge provided by artificial intelligence. Moreover, the use of deep reinforcement learning in algorithmic trading has expanded, with DRL agents now integral to developing autonomous trading systems that not only predict prices but also generate trading signals.

This paper aims to equip researchers with the tools to understand and replicate existing studies, fostering continued advancement in the field. It specifically details the application of ML, DL, RL, and DRL in the context of pair trading within quantitative trading and the broader stock market. Furthermore, it identifies and discusses prospective research directions that build upon the foundational insights previously described.

Keywords

Pair Trading, Machine Learning, Deep Learning, Reinforcement Learning, Artificial Intelligence, Quantitative Trading

TABLE OF CONTENTS

1. Introduction

The endeavor to forecast the future valuation of company shares or other financial assets traded on exchanges is known as stock market forecasting. A trader adept in predicting future stock prices can secure significant gains. The efficient market hypothesis posits that stock prices already incorporate all available information, suggesting that price movements not rooted in new information are fundamentally unpredictable. The effectiveness of traditional prediction models, which rely solely on historical stock data, is hampered by the stock market's acute sensitivity to external information. However, the widespread adoption of the Internet has ushered in novel forms of collective intelligence, such as Google Trends and Wikipedia, whose modifications can significantly influence stock market dynamics. It is believed that stock prices are influenced not only by the sentiment surrounding financial news but also by security prices. The task of predicting stock market trends has garnered extensive interest from both the academic and business sectors. However, the question remains: To what extent can historical stock price data be utilized to forecast future stock prices?

Early investigations into the capability to forecast stock market movements heavily leaned on concepts like the efficient market hypothesis (EMH) and the theory of random walks. These foundational theories posited that stock prices move unpredictably in response to new information, making their future prices impossible to predict based on past or current data. As such, the likelihood of correctly forecasting stock price movements was deemed to be no better than chance, at a mere 50%. Nevertheless, an expanding corpus of research is now challenging these assertions, suggesting that stock markets exhibit discernible patterns that allow for some level of prediction, thereby questioning the fundamental tenets of the EMH. The consistent outperformance of benchmarks like the S&P index by investors such as Warren Buffet is often pointed to by industry professionals as tangible evidence against the EMH, indicating that stock market prediction is not only possible but practicable.

Constructing a reliable model for stock price prediction is challenging due to the multitude of variables that can influence stock prices. These variables range from breaking news and social media sentiment to fundamental analysis, company performance metrics, historical price trends, and the impact of national economic indicators like government bond prices. A model that focuses solely on one factor is likely to miss the mark. Therefore, incorporating a diverse array of factors—such as current news, social media trends, and past price data—can significantly enhance the model's accuracy. By considering multiple influences, the predictive model becomes more robust, offering a more nuanced understanding of potential price movements.

Figure 1 outlines the diverse implementations of AI in the domain of stock market transactions. It delves into quantitative trading, which leverages computer algorithms and various mathematical models, ranging from basic to complex, aimed at identifying and capitalizing on market opportunities. It also covers algorithmic trading, which automates the execution of trades based on specific criteria such as time, price through pre-set trading instructions. Furthermore, the figure highlights High-Frequency Trading (HFT), a strategy that utilizes sophisticated algorithms to conduct a vast number of transactions within mere fractions of a second. Additionally, it explains an Automated Trading System (ATS), a subset of algorithmic trading, which uses software to automatically place buy and sell orders and forward them to either a trading venue or exchange.

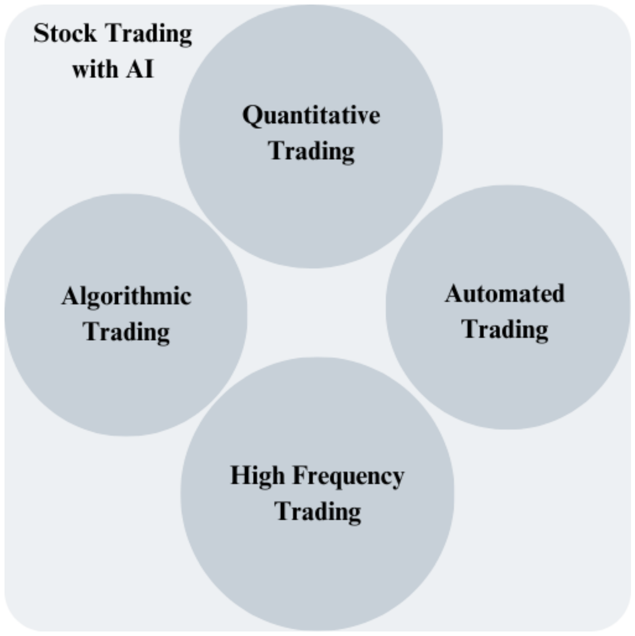


Figure 1. Utilization of Artificial Intelligence (AI) Technologies in Quantitative Finance and Equity Markets Trading

Figure 2 outlines various methodologies employed in algorithmic trading. The trend-following strategy capitalizes on an asset's upward or downward price movements by buying during an uptrend and selling during a downtrend, based on the expectation that these trends will continue. Mean reversion, in contrast, operates on the principle that asset prices and historical returns tend to revert to their historical averages over time. Momentum trading focuses on exploiting the continuation of existing price trends in asset purchases and sales. Statistical arbitrage involves constructing large, diversified portfolios to exploit short-term mispricings between related financial instruments. This strategy uses advanced quantitative models to identify and capitalize on statistical relationships and mean-reverting behaviors among securities, often employing high-frequency trading to execute trades swiftly and adjust positions. Techniques such as co-integration and regression analysis are commonly used to detect when related asset prices diverge from their expected relationships, with the aim of profiting from their eventual convergence. The moving average crossover strategy identifies potential market support or resistance levels by calculating average prices over specified periods using historical data, providing signals that reflect past price behaviors rather than predictive certainty. Lastly, the breakout strategy focuses on recognizing price levels or zones that an asset has historically struggled to surpass, and it capitalizes on the expectation that the asset's price will exceed these thresholds, a phenomenon known as a "breakout." This refined explanation provides a clearer understanding of the range of strategies used in algorithmic trading, particularly emphasizing the detailed mechanisms of statistical arbitrage.

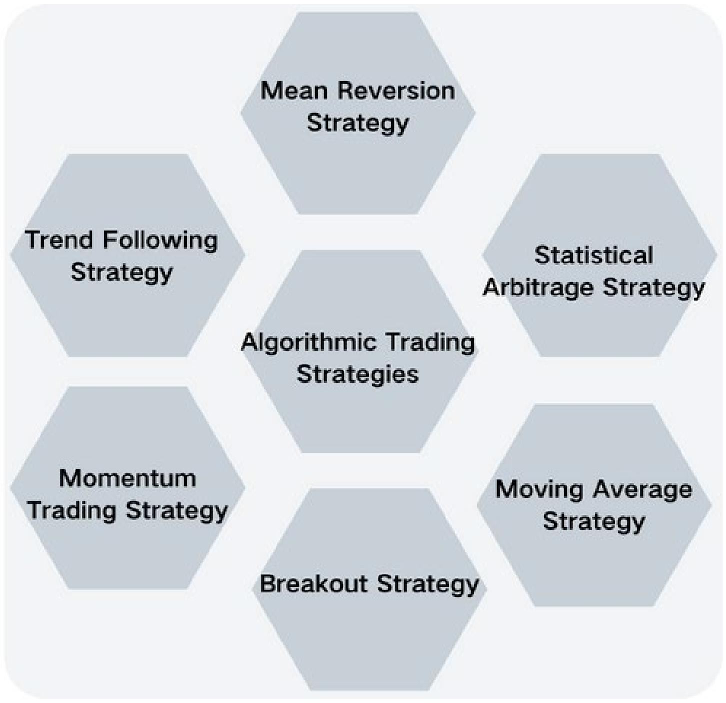


Figure 2. Diverse Approaches to AI-Enhanced Algorithmic Trading Strategies

Pair trading is a type of statistical arbitrage strategy outlined in the above figure. This approach involves identifying pairs of stocks whose prices have historically moved together and are expected to continue doing so. When the spread between the stock prices diverges beyond a predefined threshold, a trade is executed with the expectation that the spread will eventually revert to its historical mean. This strategy, while conceptually simple, poses significant challenges in the selection of stock pairs, the determination of thresholds, and the timing of trade execution.

The application of machine learning (ML), deep learning (DL), and reinforcement learning (RL) techniques has the potential to substantially enhance the effectiveness of pair trading strategies. ML algorithms can process vast amounts of market data to identify complex patterns and correlations between securities that may not be apparent through traditional statistical methods. For instance, supervised learning models can be trained on historical data to forecast price movements or to classify potential pairs based on the likelihood of their spreads reverting.

Deep learning, a subset of ML characterized by neural networks with multiple layers, can capture deeper levels of abstraction in data. In the context of pair trading, DL models, particularly those using recurrent neural networks (RNNs) or long short-term memory (LSTM) networks, are adept at analyzing time-series data and can be instrumental in predicting the future behavior of the spread between a pair of stocks.

Reinforcement learning, a type of ML where agents learn to make decisions by performing actions and receiving feedback from the environment, can be used to optimize trade execution.

RL agents can learn from market interactions to make sequential decisions, adjusting trading actions based on the evolving state of the market and the performance of the trading strategy.

Overall, these advanced analytical techniques offer promising improvements to pair trading strategies, allowing for more dynamic and adaptive models that can learn from data and refine their predictions over time. As financial markets become increasingly complex and data-driven, leveraging ML, DL, and RL in pair trading could provide a competitive edge to traders and financial institutions.

The primary objective of this literature review is to systematically examine and synthesize existing approximately 100 papers over the past decade papers on the application of machine learning (ML), deep learning (DL), and reinforcement learning (RL) in the context of pair trading strategies. It aims to:

1. Identify the Current State of Research: Map out the current landscape of methodologies and techniques employed in the domain of pair trading, with a particular focus on how ML, DL, and RL have been integrated into these strategies.

2. Evaluate Methodological Approaches: Critically assess the various algorithmic and computational approaches that have been used, comparing their effectiveness, efficiency, and scalability.

3. Highlight Innovations and Advances: Detail the innovative applications and advances in the field, especially those that have shown significant improvements over traditional statistical methods.

4. Understand Performance Metrics: Analyze the performance metrics used in existing studies to evaluate the success of trading strategies, providing insights into the risk and return profiles of such applications.

5. Explore Challenges and Limitations: Discuss the challenges and limitations faced by researchers and practitioners in applying ML, DL, and RL to pair trading, including data quality, computational demands, and model overfitting.

6. Recommend Best Practices: Based on the review, recommend best practices for integrating these learning techniques into pair trading strategies, suggesting frameworks or methodologies that have demonstrated success.

7. Propose Future Research Directions: Identify gaps in the current literature and propose directions for future research that could further improve pair trading strategies through the use of advanced learning techniques.

This survey's principal contributions are articulated below:

1. We delve into the latest advancements within the realm of artificial intelligence, focusing particularly on innovations that have surfaced over the past ten years.

2. The study scrutinizes the applications of Machine Learning, Deep Learning, and Reinforcement Learning in the realm of pair trading on stock market forecasting, reflecting on developments over the preceding decade.

3. An exploration of the stock market's overarching mechanisms is undertaken, drawing upon existing scholarly work.

4. Directions for future investigations in the domain of stock market predictions are proposed, aiming to guide emerging scholars towards novel research avenues.

5. Furthermore, we outline potential data sources that researchers can leverage for their investigations.

The structure of the remainder of this document is organized as follows: Section 2 elaborates on the research works that inform this study. Section 3 provides a concise introduction to the scope of this survey. Section 4 discusses the methodologies for data processing and feature extraction. In Section 5, we offer a detailed examination of the various forecasting techniques, enriched with relevant context. Section 6 focuses on the metrics used to assess prediction accuracy. The framework for implementation and the availability of data are discussed in Section 7. Section 8 envisions the prospective avenues for further inquiry in this field. The paper concludes with a summary in Section 9, encapsulating the essence of the study, and final thoughts are presented in Section 10.

2. Related Work

In this section, I delve into the diverse applications of machine learning methodologies within the realm of pair trading, exploring how these computational techniques have been tailored to enhance trading strategies. This section is structured to reflect the four main categories of learning paradigms that have been influential in shaping pair trading models.

This literature review encapsulates the progressive integration of machine learning into pairs trading, showcasing the shift from traditional approaches to advanced computational models. Initially focusing on unsupervised learning for identifying trading pairs based on market data, the field has evolved to incorporate supervised learning techniques for refining trade timing and selection. The advent of deep learning has further advanced the strategy, enabling the analysis of complex patterns for improved decision-making. Reinforcement learning represents the latest frontier, offering dynamic strategy adjustments to optimize returns. Across these developments, machine learning has proven pivotal in enhancing the efficacy and profitability of pairs trading, demonstrating its critical role in the future of financial market strategies.

2.1 Unsupervised Learning Methods

This section begins with an exploration of unsupervised learning methods, where algorithms identify patterns and relationships in data without requiring labeled outcomes. These methods are crucial for detecting naturally occurring correlations in financial markets and have significantly contributed to the development of pairs trading strategies.

The selection of studies for this review was guided by specific criteria to ensure relevance and rigor. First, the papers chosen had to directly address the application of unsupervised learning techniques in financial trading, particularly in the context of pairs trading and statistical arbitrage. Second, the studies were required to demonstrate empirical testing using real-world market data, ensuring that the findings are grounded in practical applications. Finally, preference was given to research that explores innovative or advanced methodologies, such as machine learning algorithms, which have the potential to enhance traditional trading strategies.

Yuxing Chen, Weiluo Ren, and Xiaoxiong Lu (2012) offers a groundbreaking examination of how machine learning can enhance pairs trading, a statistical arbitrage technique aimed at exploiting mean-reverting spreads between paired securities. The study meticulously explores two distinct approaches: a straightforward Portfolio Rebalancing & Linear Regression method and a more complex Kalman Filter and Expectation Maximization (EM) Algorithm technique, each designed to predict and capitalize on price movements within the China futures market. Their findings suggest that while both methodologies hold promise, the application of advanced machine learning techniques like the Kalman Filter could significantly boost the profitability of pairs trading strategies, thereby merging traditional financial strategies with cutting-edge computational methods.

Muhammad Khalid Sohail et al. (2020) explores the utilization of DBSCAN clustering alongside traditional pair trading techniques to identify clusters from market-cap sector classification and BVS with PCA on daily data returns of sample firms. Implementing machine learning, the study achieves a remarkable average excess monthly return, validating mean reversion and market neutrality at Pakistan Stock Exchange (PSX). This research highlights how machine learning can enhance profitability in pair trading strategies, offering new insights into the financial market's dynamics and investment opportunities within the Pakistan Stock Exchange.

The study by Muhammad Khalid Sohail et al. (2022) investigates the effectiveness of pair trading strategies during the COVID-19 pandemic using an unsupervised machine learning approach. By employing the DBSCAN algorithm to form trading pairs from market and accounting data, the research seeks to determine if pair trading can still be profitable amid the market volatility and uncertainty caused by the pandemic. This work adds to the existing literature by examining the resilience of pair trading strategies in a significantly disrupted market environment.

Chulwoo Han (2023) introduces a novel approach to pairs trading by leveraging unsupervised learning, marking a significant departure from traditional pairs trading strategies that rely primarily on return time series or cointegration methods. By incorporating firm characteristics alongside price data, the study uncovers pairs with greater potential for co-movement, leading to a more effective trading strategy. The approach is validated through application to the US stock market from 1980 to 2020, showcasing the strategy's robust performance and the utility of firm characteristics in identifying trading pairs.

In summary, the reviewed studies highlight the transformative impact of unsupervised learning methods on pairs trading strategies in financial markets. These approaches enable the identification of complex patterns and relationships without labeled data, enhancing the effectiveness of statistical arbitrage by incorporating advanced techniques such as machine learning, clustering, and regression. The research demonstrates that integrating innovative methodologies, like the Kalman Filter, DBSCAN clustering, and firm-specific characteristics, can significantly improve the profitability and resilience of pairs trading, especially in volatile and disrupted market conditions. Collectively, these studies underscore the potential of unsupervised learning to refine traditional financial strategies and adapt to dynamic market environments, offering new insights and opportunities for algorithmic trading.

2.2 Supervised Learning Methods

Next, we review supervised learning methods, where models are trained on historical data with known outcomes to forecast future events. This approach is particularly relevant for pair trading, where predictive accuracy is paramount. Studies within this domain have contributed significantly to the refinement of entry and exit points for trades, aiming to optimize the timing and selection of paired assets.

In the research by Gopal Rao Madhavaram (2013), an innovative approach combining dynamic Principal Component Analysis (PCA) and Support Vector Machines (SVM) for financial market analysis is explored. Focusing on the Financial Select Sector SPDR Fund (XLF), the study applies PCA to distill systematic risk factors from stock data and validates trading strategies using SVM. The findings indicate that SVM can optimize trading decisions in certain scenarios, showcasing the potential of integrating machine learning with traditional financial analysis techniques for enhanced algorithmic trading strategies.

In the study by Jarley P. Nóbrega and Adriano L. I. Oliveira (2013), the authors delve into enhancing statistical arbitrage strategies for intraday trading. They employ a unique combination of Extreme Learning Machine (ELM) and Support Vector Regression (SVR) models alongside linear regression techniques, focusing on pairs trading within the financial sector. The integration of the traditional Kalman Filter aims to refine the statistical performance of individual forecasts from the ELM and SVR models. Through the analysis of cointegrated pairs, the research presents evidence that financial performance can be significantly improved using at least one linear combination technique. This paper not only contributes to the existing body of knowledge by introducing a novel approach for statistical arbitrage but also highlights the potential for combining machine learning methods to enhance trading strategies within the volatile domain of intraday trading.

A same approach to financial forecasting is explored by Jarley P. Nóbrega and Adriano L. I. Oliveira (2014) again in the next year, combining Extreme Learning Machine (ELM) and Support Vector Regression (SVR) with Kalman filter regression models. This research evaluates the effectiveness of these combined forecasting models in enhancing the performance of statistical arbitrage strategies in financial markets. The study demonstrates that integrating machine learning models with a Kalman filter significantly improves the accuracy of financial forecasts, impacting positively on the annualized returns and reducing volatility.

Jiayu Wu (2015) presents a novel approach to pairs trading by utilizing a combination of the spread model, Ornstein-Uhlenbeck (O-U) model, and Support Vector Machines (SVM) on the stocks GOOG/GOOGL. The study is unique in its inclusion of technical indicators' spreads, as well as its use of two new metrics aimed at predicting future prices rather than returns, demonstrating a successful strategy with a good win-rate.

Tamal Datta Chaudhuri et al. (2017) explore the efficacy of machine learning algorithms, specifically Support Vector Machine (SVM), Random Forest (RF), and Adaptive Neuro Fuzzy Inference System (ANFIS), for predictive modeling in pairs trading. This innovative approach focuses on forecasting the price ratio of company pairs using a set of nine selected features. The study highlights the successful application of these algorithms to predict market movements and suggests that machine learning can significantly enhance trading strategies through its predictive accuracy, providing a new avenue for financial market analysis and investment strategy development.

Ian Sutherland et al. (2018) investigates the application of various machine learning models on the KOSPI 200 for statistical arbitrage. Through a comparative analysis between classification and prediction models, including logistic regression, random forest, deep neural networks, and gradient-boosted trees, the study reveals that all tested models significantly outperformed the KOSPI 200 index. The research highlights the effectiveness of machine learning in generating profitable trading signals in the South Korean stock market, with classification models slightly outperforming prediction models.

The study by Jifeng Sun et al. (2019) explores the application of Random Forest algorithms to predict cryptocurrency prices, utilizing factors from Alpha101 on market data from Binance and Bitfinex. The results demonstrate the efficacy of their strategy in cryptocurrency trading, highlighting the potential of machine learning techniques in navigating the volatile cryptocurrency market. This research contributes to the understanding of predictive modeling in financial markets, specifically within the rapidly evolving domain of cryptocurrencies.

The study by Sungju Hong and Soosung Hwang (2023) explores the effectiveness of using firm characteristics to identify pairs for trading, addressing the challenge of spurious pair identification through multiple hypothesis testing. They found that pairs with higher firm characteristic similarities yield better performance by minimizing non-convergence risk. Despite a general decline in pairs trading profitability post-2003, their findings suggest that incorporating firm fundamentals can refine pair selection and enhance trading strategies, especially during market crises where firm fundamentals become even more critical.

In summary, the reviewed studies on supervised learning methods demonstrate their significant potential in enhancing pairs trading strategies through predictive modeling. By leveraging historical data with known outcomes, these approaches refine the timing and selection of paired assets, optimizing entry and exit points for trades. Key methodologies include the use of dynamic Principal Component Analysis (PCA), Support Vector Machines (SVM), Extreme Learning Machine (ELM), Support Vector Regression (SVR), Kalman filters, and other machine learning techniques like Random Forest (RF) and Adaptive Neuro Fuzzy Inference System (ANFIS).

These studies collectively show that integrating machine learning models with traditional financial analysis not only improves predictive accuracy but also enhances trading performance across various markets, including stocks, futures, and cryptocurrencies. Furthermore, the research highlights the importance of incorporating firm characteristics and addressing challenges such as spurious pair identification, suggesting that a deeper understanding of firm fundamentals can refine pairs trading strategies and bolster profitability, especially in volatile or crisis-prone market conditions.

2.3 Deep Learning Methods

Following that, the review covers deep learning methods, a subset of machine learning characterized by complex neural networks capable of uncovering intricate patterns in large datasets. In the context of pair trading, deep learning has been a game-changer, providing nuanced insights into market dynamics and facilitating the development of more advanced trading algorithms.

It should be noted that this article does not address deep reinforcement learning as a standalone topic but rather includes it within the broader frameworks of deep learning and reinforcement learning. Therefore, it is essential to clarify the relationship between these three areas. Deep Reinforcement Learning (DRL) is a subset of machine learning that integrates the principles of deep learning and reinforcement learning to address complex decision-making problems. Reinforcement Learning (RL) is a type of machine learning where an agent learns optimal decision-making by interacting with an environment, receiving feedback in the form of rewards or penalties, and refining its actions to maximize cumulative rewards over time. Deep Learning (DL), on the other hand, employs deep neural networks with multiple layers to model intricate patterns and representations in data, automatically extracting features from raw inputs, making it highly effective for tasks such as image recognition and natural language processing. DRL combines these approaches by utilizing deep neural networks within reinforcement learning frameworks to approximate value functions, policies, or models. This fusion allows DRL to handle high-dimensional inputs, such as images or raw sensor data, enabling it to learn directly from these inputs and making it suitable for complex applications like robotics, game playing, and autonomous driving. By leveraging deep learning's capacity for representation learning, DRL enables reinforcement learning algorithms to scale to problems with large state and action spaces, which would be impractical for traditional RL methods alone. Thus, DRL effectively harnesses the strengths of both deep learning and reinforcement learning, enhancing its capability for sequential decision-making in complex environments.

Christopher Krauss et al. (2017) examines the application of advanced machine learning techniques—deep neural networks (DNNs), gradient-boosted trees (GBTs), and random forests (RFs)—to statistical arbitrage strategies in the S&P 500. By forecasting the probability of stocks outperforming the market, the study demonstrates the effectiveness of these models, especially when combined into ensembles, in generating significant out-of-sample returns. The findings not only highlight the potential for machine learning in enhancing trading strategies but also challenge the semi-strong form of market efficiency, with the ensemble model showing particularly promising results.

Gheorghe Ruxanda and Sorin Opincariu (2018) introduce a sophisticated approach that integrates Bayesian neural networks (BNNs) with Dirichlet process mixtures for modeling pairs trading strategies. The paper showcases how this hierarchical model, with priors derived from a Dirichlet process mixture model, allows for dynamic adjustment to non-stationarity in financial data, significantly enhancing the adaptability and predictive power of pairs trading strategies. Through an application to pairs trading, the research illustrates the model's ability to capture complex relationships between financial assets, providing a more nuanced and effective tool for navigating financial markets.

The paper by Andrew Brim (2019) employs deep Q-networks (DQN) to enhance pairs trading strategies within the stock market, particularly testing on 38 cointegrated stock pairs. This approach leverages reinforcement learning to predict and exploit mean reversion in stock pair spreads, demonstrating the DQN's capacity to consistently produce positive returns. By training on data from 2014-2017 and testing on 2018 data, the study illustrates the potential of deep reinforcement learning in financial markets, highlighting its ability to adapt and learn from complex market dynamics for profit maximization.

Shen-Hang Huang et al. (2020) present a novel approach for detecting structural breaks in the stock market, focusing on pairs trading. The study introduces a hybrid deep learning model that utilizes both Wavelet Transform for frequency-domain feature extraction and a combination of convolutional neural networks (CNN) and long short-term memory (LSTM) networks for time-domain analysis. The model outperforms traditional methods in detecting structural breaks, showcasing its potential to enhance pairs trading strategies by leveraging minute-scale stock data from the Taiwan Stock Exchange. This advancement suggests a significant step forward in applying machine learning techniques to complex financial market dynamics.

The paper by Andrea Flori and Daniele Regoli (2021) explores the application of deep learning, specifically Long Short-Term Memory (LSTM) networks, to identify pairs-trading opportunities in the stock market. The authors focus on the reversal effect, where market deviations are expected to correct over time, offering profitable trading signals. By comparing and combining LSTM predictions with traditional trading practices based on price or returns gaps, the study aims to improve portfolio performance under various investment scenarios. The analysis confirms that strategies incorporating LSTM predictions can enhance financial performance, providing valuable signals beyond those captured by price and returns gaps.

Tara Relan (2021) investigates the predictive power of the USD/GBP exchange rate on the FTSE100 index using a Temporal Convolutional Network (TCN), achieving an accuracy of 89.96%. The research spans from December 31, 1985, to October 6, 2021, employing 13,028 data points to develop a machine learning model and subsequently testing two trading strategies: pairs trading using Bollinger Bands and a buy and hold strategy. The pairs trading strategy notably outperformed the buy and hold strategy in terms of Annual Average Return and Sharpe

Ratio, despite a higher Maximum Drawdown, indicating a successful application of machine learning techniques in predicting stock market movements based on currency exchange rates.

Juan Du (2022) delves into mean-variance portfolio optimization leveraging deep learning forecasts for stocks exhibiting cointegration, showcasing a novel approach in financial market analysis. This research pioneers the integration of advanced deep learning models to enhance forecast accuracy and subsequently inform portfolio allocation decisions. By focusing on cointegrated stocks, the study not only adheres to the classical portfolio theory but also embraces modern computational techniques, marking a significant stride in the application of AI in finance. The methodology's strength lies in its ability to dynamically adjust to market conditions, potentially offering superior returns on investment.

The paper by Federico Platania et al. (2023) presents an innovative approach to pair trading by integrating multi-objective programming, cyclical insights, and neural networks. This strategy aims to exploit market inefficiencies through statistical arbitrage, enhancing the traditional pair trading framework by incorporating a broader range of analytical tools. By analyzing cyclical behaviors and employing neural networks for predictive accuracy, the approach seeks to optimize trading performance by balancing profitability with risk management.

In summary, the reviewed studies on deep learning methods highlight the significant advancements in pairs trading and statistical arbitrage strategies facilitated by complex neural networks and deep reinforcement learning. These approaches leverage deep learning's ability to uncover intricate patterns in large datasets, enhancing predictive accuracy and trading performance. Key methodologies include the use of deep neural networks (DNNs), convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and reinforcement learning models like deep Q-networks (DQN), which are employed to forecast market dynamics, detect structural breaks, and optimize trading decisions. The integration of these advanced techniques not only improves the profitability and adaptability of pairs trading strategies but also challenges traditional market efficiency theories by consistently identifying trading opportunities in complex and high-dimensional environments. Collectively, these studies underscore the transformative potential of deep learning in refining financial strategies and navigating the evolving landscape of algorithmic trading.

2.4 Reinforcement Learning Methods

Finally, we assess the role of reinforcement learning methods, where agents learn optimal behaviors through interactions with a dynamic environment. This segment of the review will emphasize how reinforcement learning has been applied to the decision-making processes in pair trading, focusing on the strategies that have successfully navigated the trade-offs between exploration and exploitation to maximize returns.

Saeid Fallahpour et al. (2016) explores an innovative approach to optimize pairs trading strategies through reinforcement learning. By dynamically adjusting trading parameters based on market conditions, the study demonstrates significant improvements in trading performance compared to traditional methods. Utilizing cointegration and reinforcement learning, the research offers a novel perspective on maximizing profits and managing risks in pairs trading strategies, showcasing its efficiency with empirical data from S&P 500 constituent stocks.

Andrew Brim (2020) explores the innovative application of Double Deep Q-Networks (DDQNs) to pairs trading, leveraging the mean reversion property of stock prices for profit. By training a DDQN model on historical stock pair data, the study demonstrates the model's ability to predict profitable trading signals in the stock market, emphasizing the utility of reinforcement learning in financial strategies. This research is notable for introducing a Negative Rewards Multiplier (NRM) to adjust the model's risk-taking behavior, illustrating a significant advance in applying deep reinforcement learning to complex market dynamics.

Georgios Sermpinis (2020) introduces a pioneering Cointegration Approach-Deep Reinforcement Learning (CA-DRL) framework to execute pairs trading in commodities markets. Leveraging both cointegration to select commodity pairs and deep reinforcement learning (DRL) for trading decision-making, the CA-DRL model outperforms traditional pairs trading strategies in terms of returns, with comparable risk levels. The inclusion of a genetic algorithm further optimizes the trading thresholds, showing a slight improvement over standard method. This innovative combination underscores the potential of integrating advanced machine learning techniques with traditional financial strategies to enhance trading performance.

Xiangyu Zong et al. (2021) introduces a novel Co-integration method and Deep Reinforcement Learning (CA-DRL) model applied to pairs trading strategy on the Dalian Commodity Exchange (DCE) for soybean futures and its derivatives. It demonstrates the superior performance of the CA-DRL model over traditional models, whether the pair formation period is one, two, or three years, highlighting its effectiveness in generating profitable trading strategies between commodities.

Sang-Ho Kim et al. (2022) introduces HDRL-Trader, a hybrid deep reinforcement learning method for pairs trading that optimizes both trading actions and stop-loss boundaries using two independent reinforcement learning networks. By employing novel techniques like dimensionality reduction, clustering, regression, and behavior cloning, HDRL-Trader significantly outperformed state-of-the-art models in the S&P 500 index, demonstrating a 25.7% higher average return rate. This approach offers a new pathway for algorithmic trading strategies by integrating the strengths of TD3 and DDQN algorithms within a hybrid framework.

Jing-You Lu (2022) introduces a two-phase machine learning framework, SAPT, designed to optimize pairs trading strategies by incorporating structural break detection. The first phase employs a hybrid model to detect structural breaks, while the second phase focuses on optimizing the trading strategy, taking into account transaction costs and market-closing risks. By leveraging machine learning techniques, SAPT significantly outperforms existing strategies in the Taiwan stock market, demonstrating the potential of integrating structural break detection into pairs trading strategies for enhanced profitability and risk management.

Zhizhao Xu and Chao Luo (2023) introduce an enhanced pairs trading strategy leveraging a two-level reinforcement learning framework, which amalgamates pair selection via the Extended Option-Critic (EOC) method and trade thresholds setting through a Multi-Agent Deep

Deterministic Policy Gradient (MADDPG) approach. This novel framework outperforms traditional methods by dynamically selecting trading pairs and optimizing trade thresholds, demonstrating superior returns in the Chinese futures market. The research signifies a pivotal shift in pairs trading strategies by integrating deep reinforcement learning to address the complexity of financial markets.

In summary, the reviewed studies on reinforcement learning methods showcase their transformative potential in optimizing pairs trading strategies through dynamic decision-making and adaptive learning. Reinforcement learning enables trading agents to balance exploration and exploitation, thereby maximizing returns in complex and volatile markets. Key approaches include the application of advanced models such as Double Deep Q-Networks (DDQNs), Cointegration Approach-Deep Reinforcement Learning (CA-DRL) frameworks, and hybrid deep reinforcement learning methods like HDRL-Trader, which combine various reinforcement learning algorithms to optimize trading actions and risk management. These studies demonstrate that reinforcement learning can significantly enhance pairs trading by dynamically adjusting trading parameters, selecting optimal pairs, and setting effective trade thresholds based on real-time market conditions. The integration of reinforcement learning with traditional financial techniques, such as cointegration and structural break detection, further refines trading strategies, resulting in improved profitability and resilience across diverse financial markets, including stocks, commodities, and futures. Collectively, these advancements underscore the evolving role of reinforcement learning in modern algorithmic trading, providing a robust framework for navigating the complexities of financial markets.

3. Landscape Overview

Within this section, we delve into the central document which anchors our inquiry in this research. Sourced through extensive database queries, this paper, along with additional supporting documents, has been curated from renowned academic sources, including but not limited to Google Scholar. The research sweep employed several scientific repositories such as Scopus, Google Scholar, Springer, IEEE Xplore, Science Direct, and Web of Science to ensure a comprehensive collection of literature. Keywords deployed in the search strategy encompassed a range of terms from "Pair Trading with Deep Learning", “Pair Trading with Reinforcement Learning”, “Pair Trading with Deep Reinforcement Learning” to "Pair Trading with Machine Learning", “Pair Trading with Supervised Learning”, “Pair Trading with Unsupervised Learning,” aiming to capture the breadth of machine learning's role in financial decision-making. Papers that did not directly contribute to the crux of our thematic investigation were omitted. An accompanying **Table 1**, as follow, delineates the distribution of articles reviewed, year by year, complete with reference citations for each item.

Table 1. Counts of publication frequencies in the studies over the years analyzed, from 2012 to 2023

Year	Count	Article

2023	12	[1-12]
2022	18	[13-30]
2021	8	[31-38]
2020	14	[39-52]
2019	3	[53-55]
2018	3	[56-58]
2017	4	[59-62]
2016	1	[63]
2015	1	[64]
2014	1	[65]
2013	2	[66, 67]
2012	1	[68]

Table 2 describes of the Integration of Artificial Intelligence in Pair Trading: This table encapsulates the various elements involved in the application of AI to stock market operations, including the Categories of Stock Market Data, Types of Machine Learning Applications, Empirical and Prediction Models, Model Performance Evaluation Metrics, and Measures for Evaluating Portfolio Outcomes.

Among Machine Learning models, RF stands for Random Forest, a versatile and widely-used ensemble method. AdaBoost refers to Adaptive Boosting, a technique that combines multiple weak learners into a stronger model. DT is the abbreviation for Decision Tree, a fundamental classification and regression approach. Boosting is a family of algorithms that enhance the performance of machine learning models. LM denotes Linear Model, often used in the context of regression, or Lasso Model when it involves L1 regularization. LR stands for Logistic Regression, a staple method for binary classification tasks. SVM and SVR represent Support Vector Machine and Support Vector Regression, respectively, both of which are powerful in finding the optimal boundary between different classes. NB is short for Naive Bayes, a probabilistic classifier that assumes independence between predictors. EN means Elastic Net, a regularized regression method that combines both L1 and L2 penalties. GDA stands for Gaussian Discriminant Analysis, a generative model for classification, while ELM is the acronym for Extreme Learning Machine, a fast-learning algorithm for single-hidden layer feedforward neural networks.

In Deep Learning, NN is the abbreviation for Neural Network, a basic structure of interconnected nodes mimicking the human brain. DNN stands for Deep Neural Network, which is a complex network with multiple layers, enabling the model to learn high-level features from data. RNN, or Recurrent Neural Network, is a type of neural network where connections between nodes form a directed graph along a temporal sequence, allowing it to exhibit temporal dynamic behavior. LSTM is Long Short-Term Memory Network, a special kind of RNN capable of learning long-term dependencies. CNN represents Convolutional Neural Network, highly effective in processing data with a grid-like topology, such as images. TCN stands for Temporal Convolutional Network, known for its use in sequence modeling tasks.

For Reinforcement Learning, DQN is used for Deep Q-Network, a groundbreaking algorithm that combines Q-Learning with deep neural networks. DDQN, or Double Deep Q-Network, is an improvement over DQN that reduces overestimation of action values. PPO refers to Proximal Policy Optimization, a policy gradient method for training deep neural networks. HDRL stands for Hierarchical Deep Reinforcement Learning, which structures the learning process in a hierarchical fashion for complex tasks. Lastly, DPG is used for Deep Policy Gradient, often called Deep Deterministic Policy Gradient when it is used in a context of continuous action spaces.

Table 2. The Categories of Stock Market Data, Types of Machine Learning Applications, Empirical and Prediction Models, Model Performance Evaluation Metrics, and Measures for Evaluating Portfolio Outcomes

Types of Data	Tasks	Models[1]	Model Performance Evaluation Metrics	Portfolio Performance Evaluation Metrics
Company Stocks	Empirically investigate the pairs trading performance	Machine Learning (RF, AdaBoost, DT, Boosting, LM, LR, SVM, SVR, NB, EN, GDA, ELM)	MAE	Cumulative Return
Stock Index	Price prediction	Deep Learning (NN, DNN, RNN, LSTM, CNN, TCN)	MAPE	Daily Return
Stock Index Futures		Reinforcement Learning (DQN, DDQN, PPO, HDRL, DPG)	MSE	Monthly Return
Commodities			RMSE	Average Daily Return
Cryptocurrencies			Confusion Matrix	
Currencies			Accuracy	Maximum Drawdown
			Precision	Skewness
			Recall	Kurtosis
			F-Score	Sharpe Ratio
			Model Loss	Treynor ratio
			AUC	Profit-loss ratio
			ROC Curve	Annual Volatility
			Reward	Standard Deviation

[1] A detailed introduction of the method will be provided in Chapter 5.

In our exploration of recent advancements within the field, we undertook a detailed review of over seventy academic papers to identify trends in methodological approaches among researchers. Our findings indicate a notable diversification in the tools and models employed: approximately 20% of scholars favored the exclusive use of reinforcement learning, mirroring the percentage that opted solely for deep learning techniques. The predominant approach, however, was traditional machine learning, which was utilized by nearly half of the researchers, accounting for 48% of the study sample. A smaller fraction, about 10%, combined multiple methodologies, indicating a trend towards integrative or hybrid models.

The scope of our analysis encompassed key financial indices across several regions, providing a global perspective on market behaviors. For the United States, we included major indices such as the S&P 500, Russell 2000 Index, New York Stock Exchange (NYSE) Composite, and American Stock Exchange (AMEX) Composite. Our study also extended to Asian markets, examining indices like the Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX), KOSPI 200 Index, CSI 300 Composite, Pakistan Stock Exchange (PSX) Index, and Shanghai Stock Exchange (SSE) Index. European market trends were analyzed through the FTSE 100 and other European markets. Supplementary **Table 3** was prepared to detail the stock indices and their respective countries, providing a comprehensive view of the research landscape in this field.

Table 3. presenting a compilation of key stock indices and markets evaluated in this study

Index	Country
S&P 500	U.S.
S&P 400	U.S.
S&P 100	U.S.
Russell 2000	U.S.
New York Stock Exchange (NYSE)	U.S.
American Stock Exchange (AMEX)	U.S.
CSI 300	China
CSI 300 index futures (IF)	China
CSI 300 exchange traded fund (ETF)	China
Shanghai Stock Exchange (SSE)	China
Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX)	Taiwan
FTSE 100	UK
Pakistan Stock Exchange (PSX)	Pakistan
KOSPI 200	South Korea
1027 NSE (National Stock Exchange of India)	India

4. Data Processing and Analysis

Most of the studies analyzed in our review relied on daily datasets that included attributes like Opening and Closing prices, the Highest and Lowest values within the trading day, along with the Volume of trades (commonly abbreviated as OHLCV). A subset of scholars opted for more granular datasets, utilizing intraday data that captured market changes at intervals of one, five, or fifteen minutes. Additionally, there were instances where research incorporated sentiment analysis, employing textual data sourced from social media posts, such as tweets and online comments, to gauge market sentiment.

4.1 Historical Price Data with Daily Interval

Typically, historical stock data includes information on the opening price, highest price, lowest price, closing price, and the volume of shares traded for each stock. Below is an illustration of the daily dataset for Tesla, as referenced in various studies [1, 2, 5-8, 11-13, 15-23, 25-33, 35-44, 46, 49, 50, 52, 53, 56, 58-62, 65-68]. **Table 4** presents the stock market data organized by date.

In the compilation of our research, it was observed that out of the total number of papers reviewed, 51 studies specifically employed data with daily intervals in their analyses. This approach constituted a significant majority, with the adoption rate of daily interval data reaching 75%. This high usage rate underscores the prevalent preference among researchers for leveraging daily granularity in data to conduct their investigations. This trend reflects the academic community's recognition of the value in analyzing daily fluctuations and patterns within the market, which are crucial for a nuanced understanding of financial dynamics over time. Another important factor contributing to the popularity of daily data may be that higher frequency data, such as intraday or tick data, often requires a subscription or fee to access, making it less accessible for many researchers. Thus, the widespread use of daily data not only reflects methodological preference but also practical considerations regarding data availability and cost.

Table 4. An example of Tesla corporation's daily historical data

Date	**Open**	**High**	**Low**	**Close**	**Adj Close**	**Volume**
4/4/2023	197.320007	198.740005	190.320007	192.580002	192.580002	126463800
4/5/2023	190.520004	190.679993	183.759995	185.520004	185.520004	133882500
4/6/2023	183.080002	186.389999	179.740005	185.059998	185.059998	123857900
4/10/2023	179.940002	185.100006	176.110001	184.509995	184.509995	142154600
4/11/2023	186.690002	189.190002	185.649994	186.789993	186.789993	115770900
4/12/2023	190.740005	191.580002	180.309998	180.539993	180.539993	150256300
4/13/2023	182.960007	186.500000	180.940002	185.899994	185.899994	112933000
4/14/2023	183.949997	186.279999	182.009995	185.000000	185.000000	96438700
4/17/2023	186.320007	189.690002	182.690002	187.039993	187.039993	116662200
… …	… …	… …	… …	… …	… …	… …

4.2 Historical Price Data with Tick Interval

Tick historical data represent the most granular level of information available for trading activities, recording every single transaction that occurs in the stock market. Unlike the broader categories of intraday data, which aggregate price movements and trading volumes into set intervals such as minutes or hours, tick data provide a continuous stream of data points, each representing a completed trade. These data points encompass the exact time of the transaction, the price at which the trade was executed, and the size of the trade, offering an exhaustive snapshot of market activity.

The depth and precision of tick data make it an invaluable resource for various high-frequency trading (HFT) strategies, where algorithms make numerous trades over very short periods, often seconds or milliseconds. Analysts use tick data to construct detailed models of market behavior, allowing for the prediction of price movements with high accuracy. Moreover, this level of detail facilitates a deeper analysis of market liquidity, order flow, and price dynamics, essential for understanding the microstructural aspects of the market.

Tick data's comprehensive nature, however, comes with the challenge of managing vast volumes of information, necessitating robust data processing and storage capabilities. Despite these challenges, the insights garnered from tick historical data are pivotal for traders engaging in intraday trading, where strategies are refined down to the minute or second. As indicated in Table 5, which provides a tick interval stock market data of Gold Futures AU2106, as referenced in only two studies [34, 57]. The application of tick data extends beyond mere transaction recording, serving as a cornerstone for predictive models in the fast-paced environment of intraday trading.

Table 5. A sample of historical data on a tick interval basis from the Gold Futures AU2106

Contract ID	Trade Date	Previous Settle	Timestamp	Volume	Price	Bid Price	Ask Price	Last Settle	Turnover	…
au2106	20200610	20200609	21:00:00	500	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:01	0	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:01	500	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:02	0	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:02	500	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:03	0	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:03	500	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:04	0	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:04	500	391.88	391.88	391.88	391.88	391880	…
au2106	20200610	20200609	21:00:05	0	391.88	391.88	391.88	391.88	391880	…
… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	…

4.3 Historical Price Data with One-Minute Interval

One-minute interval historical data offer a balanced granularity for analyzing stock market movements, capturing price and volume information updated every minute throughout the trading session. This level of detail strikes a compromise between the exhaustive precision of tick data and the broader overview provided by longer intraday intervals. Each data point within the one-minute interval dataset includes the opening price, highest price, lowest price, closing price, and trading volume for that specific minute. This resolution is particularly beneficial for medium-frequency trading strategies, enabling traders to identify short-term trends and make decisions based on the minute-to-minute evolution of market conditions. The analysis of one-minute interval data allows for a refined examination of price volatility and trading momentum, crucial for executing strategies that capitalize on rapid market shifts. While the data volume is significantly lower than tick data, it still requires considerable processing power and sophisticated analytical tools to extract meaningful insights. The utility of one-minute interval data is demonstrated in Table 6, which showcases the one-minute interval stock market data of Tesla, as explored in a select few studies [10, 24, 48, 54, 55, 64]. This interval is particularly valuable for developing predictive models that require a balance between detail and manageability, catering to the demands of intraday trading with enhanced precision.

Table 6. An example of Tesla corporation's one-minute historical data

Datetime	Open	High	Low	Close	Adj Close	Volume
2024-04-04 09:30:00	170.0000	170.7400	169.8000	170.0303	170.0303	2425141
2024-04-04 09:31:00	170.0000	170.0301	169.3100	169.4850	169.4850	637577
2024-04-04 09:32:00	169.5050	169.6800	168.8741	168.9050	168.9050	537015
2024-04-04 09:33:00	168.8900	169.4800	168.7800	168.8800	168.8800	494559
2024-04-04 09:34:00	168.8800	168.9867	168.5200	168.5600	168.5600	466847
2024-04-04 09:35:00	168.5800	168.6350	168.2500	168.3400	168.3400	487605
2024-04-04 09:36:00	168.3875	168.7100	168.2613	168.4650	168.4650	447779
2024-04-04 09:37:00	168.4600	168.8000	168.3400	168.5400	168.5400	441173
2024-04-04 09:38:00	168.5464	168.7000	168.3201	168.4117	168.4117	308755
2024-04-04 09:39:00	168.4324	168.8500	168.3927	168.7199	168.7199	384769
… …	… …	… …	… …	… …	… …	… …

4.4 Historical Price Data with Five-Minute Interval

Five-minute interval historical data provide a condensed view of market movements, aggregating the open, high, low, close prices, and volume of stocks within five-minute blocks. This dataset is designed for traders and analysts looking for a more macroscopic perspective on intraday price dynamics without delving into the micro-level fluctuations captured by tick or one-minute data. The five-minute interval serves as an effective lens for detecting broader intraday trends, support and resistance levels, and potential breakout points, offering a pragmatic balance between detail and overview. It suits strategies that aim to exploit short-to-medium term movements, providing sufficient granularity to observe the market's rhythm while filtering out the noise of more minute-by-minute fluctuations. Despite its reduced volume compared to tick and one-minute data, five-minute interval data still necessitates advanced analytical techniques to decode the patterns and trends that drive market behavior. Highlighted in Table 7, the five-minute interval stock market data of Tesla, as mentioned in studies [3, 14, 22, 45, 54], underscores its application in crafting strategies that require a nuanced understanding of market trends over slightly extended periods. This data interval is instrumental in developing models that predict price movements with a broader temporal lens, optimizing for intraday trading strategies that operate on short to medium time frames.

Table 7. An example of Tesla corporation's five-minute historical data

Datetime	Open	High	Low	Close	Adj Close	Volume
2024-04-04 09:30:00	170.0000	170.7400	168.5200	168.5600	168.5600	4561139
2024-04-04 09:35:00	168.5800	168.8500	168.2500	168.7199	168.7199	2070081
2024-04-04 09:40:00	168.7200	169.1600	168.4500	168.7500	168.7500	1467289
2024-04-04 09:45:00	168.7342	169.6300	168.0100	169.3650	169.3650	2099000
2024-04-04 09:50:00	169.3900	169.8600	169.0300	169.4500	169.4500	1527195
2024-04-04 09:55:00	169.4300	169.5000	168.9200	169.0999	169.0999	1159740
2024-04-04 10:00:00	169.0800	169.1300	168.3736	168.7200	168.7200	1309723
2024-04-04 10:05:00	168.7600	169.7500	168.7200	169.6550	169.6550	1288829
2024-04-04 10:10:00	169.6600	170.1400	169.4300	170.0437	170.0437	1449532
2024-04-04 10:15:00	170.0274	170.1200	169.7200	169.8956	169.8956	978258
… …	… …	… …	… …	… …	… …	… …

4.5 Historical Price Data with Hourly Interval

Hourly historical data provide a macroscopic lens for analyzing the stock market, capturing the aggregate movements and trading volumes within each hour of the trading session. This interval data strikes a balance between the high-resolution tick data and the broader daily market summaries, offering a middle ground for analysis. Each data point in the hourly dataset consolidates the opening price, the peak price, the lowest price, and the closing price within the hour, along with the volume of shares traded during that time frame. Such datasets are particularly useful for traders and analysts employing strategies that operate on a shorter timeframe than daily swing trading but do not require the intensive data volume and rapid decision-making of high-frequency trading.

The hourly data sets provide a snapshot of market trends and patterns, allowing for the identification of momentum shifts and potential breakouts within the trading day. These patterns are particularly relevant for strategies that capitalize on intra-day price movements driven by news releases, economic announcements, or market sentiment. The information gleaned from hourly data, while not as exhaustive as tick data, still offers sufficient granularity to discern significant intra-day market behaviors without the overwhelming volume of tick-by-tick data.

Despite being less demanding in terms of processing power compared to tick data, hourly interval data sets still necessitate robust analytical tools to effectively interpret the data. The insights obtained from these datasets are indispensable for traders focused on hourly market dynamics, where the aim is to understand and react to the economic forces shaping market trends within each trading hour. Table 8, which is not shown here, would typically exhibit the hourly interval stock market data for a specific security such as Tesla, as has been investigated in a few studies [51, 54]. The employment of hourly data is crucial for the development of trading models that require a comprehensive yet manageable level of market detail, essential for successful intra-day trading strategies.

Table 8. An example of Tesla corporation's hourly historical data

Datetime	Open	High	Low	Close	Adj Close	Volume
2024-04-01 09:30:00	176.1600	176.3900	171.6000	171.7280	171.7280	23947033
2024-04-01 10:30:00	171.7400	172.1899	170.2100	172.1050	172.1050	15778216
2024-04-01 11:30:00	172.0900	172.4300	170.8900	172.2800	172.2800	8095873
2024-04-01 12:30:00	172.2882	173.2771	171.7400	171.8900	171.8900	7020081
2024-04-01 13:30:00	171.8600	173.1050	171.7500	172.9400	172.9400	5705376
2024-04-01 14:30:00	172.9200	174.0868	172.5901	173.9550	173.9550	7463119
2024-04-01 15:30:00	173.9500	175.2900	173.1901	175.1200	175.1200	8080117
2024-04-02 09:30:00	164.6890	167.1900	163.4300	165.5701	165.5701	45944518
2024-04-02 10:30:00	165.5800	165.8707	163.9000	165.6700	165.6700	18110912
2024-04-02 11:30:00	165.6700	166.6500	165.1129	166.2600	166.2600	11561980
… …	… …	… …	… …	… …	… …	… …

4.6 Historical Price Data with Monthly Interval

Monthly historical data aggregate the trading activity of a stock over the span of a month, providing a broad overview of market trends and performance. These data are less granular than tick or intraday data but offer valuable insights into the long-term movements of the market. Each entry within a monthly dataset typically includes the opening price at the start of the month, the highest and lowest prices reached during that month, the closing price at month's end, and the total volume of shares traded.

The comprehensive view afforded by monthly data is particularly useful for long-term investment strategies and macroeconomic analysis. Investors and analysts leverage these datasets to gauge the overall health and direction of a stock, observing patterns that may influence investment decisions over extended periods. This might include the analysis of seasonal trends, responses to quarterly earnings reports, and broader market shifts that unfold over weeks and months.

While the volume of data is significantly less than that of tick or even daily datasets, the processing and analysis of monthly data require a strategic approach that can accommodate the longer-term cycles and trends inherent to this scale. It is a valuable resource for identifying overarching market trends and for evaluating the performance of stocks against longer-term economic indicators.

Table 9, which would typically be included here, might present the monthly interval stock market data for a company like Tesla, as surveyed in select academic papers [4, 9, 47]. This type of data is particularly relevant for constructing predictive models and performing analyses that inform strategies for investors with a longer time horizon, who may be less concerned with the intricacies of daily market fluctuations and more interested in broader, strategic market positions.

Table 9. An example of Tesla corporation's monthly historical data

Datetime	Open	High	Low	Close	Adj Close	Volume
2023-05-01 00:00:00	163.1700	204.4800	158.8300	203.9300	203.9300	2681994800
2023-06-01 00:00:00	202.5900	276.9900	199.3700	261.7700	261.7700	3440477900
2023-07-01 00:00:00	276.4900	299.2900	254.1200	267.4300	267.4300	2392089000
2023-08-01 00:00:00	266.2600	266.4700	212.3600	258.0800	258.0800	2501580900
2023-09-01 00:00:00	257.2600	278.9800	234.5800	250.2200	250.2200	2439306100
2023-10-01 00:00:00	244.8100	268.9400	194.0700	200.8400	200.8400	2590570100
2023-11-01 00:00:00	204.0400	252.7500	197.8500	240.0800	240.0800	2650798400
2023-12-01 00:00:00	233.1400	265.1300	228.2000	248.4800	248.4800	2294598400
2024-01-01 00:00:00	250.0800	251.2500	180.0600	187.2900	187.2900	2343784600
2024-02-01 00:00:00	188.5000	205.6000	175.0100	201.8800	201.8800	2019907700
… …	… …	… …	… …	… …	… …	… …

In Section 4, we explored various types of historical price data with different intervals, ranging from daily to monthly, each offering unique insights into stock market behavior. Daily data, which includes attributes such as opening, closing, high, low prices, and volume (OHLCV), is the most commonly used due to its accessibility and comprehensive reflection of market dynamics within a trading day. While more granular data, such as tick data, provides exhaustive detail on every transaction, enabling precise analysis of market microstructures, it also requires significant processing power and is often less accessible due to cost barriers. Intermediate data intervals, such as one-minute, five-minute, and hourly data, offer balanced granularity, allowing analysts to capture intraday trends and volatility with manageable data volumes. These intervals are particularly suited for strategies that require timely yet comprehensive market insights, such as medium-frequency trading.

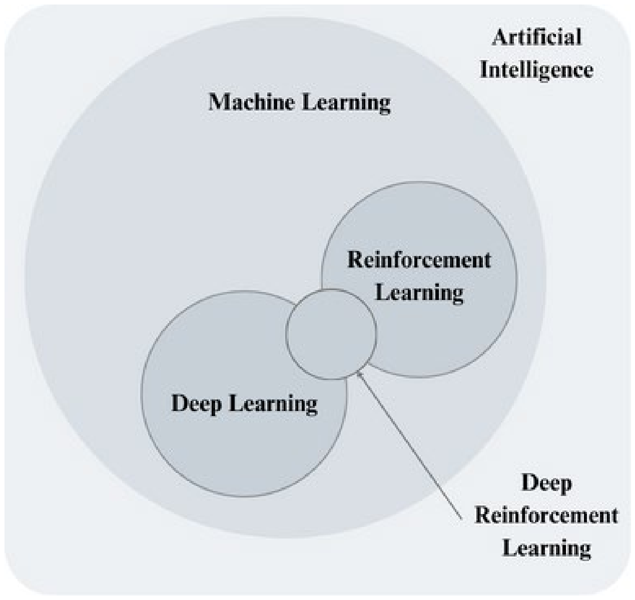
For instance, using Tesla's stock data as a reference, we observe that tick data captures the most detailed price movements, revealing every momentary change, which significantly contributes to understanding micro-level market liquidity and high-frequency trading behaviors. In contrast, daily data smooths out these fluctuations, offering a broader perspective suitable for trend analysis and longer-term investment decisions. Data with five-minute and hourly intervals serve as intermediaries, balancing detail and manageability, enabling traders to capitalize on intraday trends while avoiding the noise of tick data. Monthly data provides the most macroscopic view, capturing long-term trends and broader market shifts, essential for strategic, long-term investments.

Comparing different frequencies of Tesla's data reveals variations in moments of distribution, such as mean and variance, as well as differences in observed volatility and trends. Higher frequency data typically shows greater volatility and noise, reflecting the market's responsiveness to minute-by-minute changes. Conversely, lower frequency data smooths these fluctuations, highlighting more consistent trends over longer periods. Understanding these characteristics is crucial for selecting the appropriate data frequency for specific trading strategies, as it impacts the accuracy and effectiveness of predictive models and the overall trading outcomes. By examining Tesla's data across various intervals, we underscore the importance of choosing the right data granularity to match the trading strategy and analytical goals.

5. Artificial Intelligence Models in Pair Trading

Artificial intelligence (AI), with a specific emphasis on machine learning (ML), has garnered considerable attention and investment from the finance sector, including entities like banking institutions, fund managers, pension entities, and entities involved in securities trading. These entities prioritize the integration of various AI-driven techniques, spanning machine learning (ML), deep learning (DL), and reinforcement learning (RL), as well as methods like supervised and unsupervised learning, natural language processing (NLP), and data analytics. The goal is to refine their investment approaches, gain deeper market insights from their accumulated and acquired data, and ultimately, enhance their profitability while gaining an edge over competitors. Figure 3 illustrates the interconnections between AI, ML, DL, RL, and Deep Reinforcement Learning (DRL), highlighting that ML and DL are sub-disciplines within the broader field of AI, with DL being a further specialization within ML.

Figure 3. Relationship between AI, ML, DL, RL, and DRL. Machine learning, deep learning, and reinforcement learning all fall under the umbrella of artificial intelligence



The field of machine learning (ML) is rapidly growing and is now being applied across various industries, including quantitative trading. Quantitative trading involves the analysis of financial data using mathematical and statistical techniques. There is a growing use of machine learning techniques within quantitative trading to improve the accuracy of predictions and enhance the efficacy of financial models. Among all quantitative trading strategies, pairs trading is one of them. Specific instructions will be given in the following contexts and chapters of ML, DL, and RL in pair trading.

Pair trading is a sophisticated investment strategy utilized within the financial sector to capitalize on market efficiencies between two correlated assets. By leveraging Artificial Intelligence (AI), specifically through machine learning (ML), deep learning (DL), and reinforcement learning (RL), traders can enhance the effectiveness and responsiveness of their pair trading approaches. Here is a detailed look at how each of these technologies applies to pair trading.

Machine Learning (ML) in Pair Trading

Machine Learning is pivotal in identifying potential pairs by analyzing historical data to find two stocks or assets that exhibit a stable long-term relationship. The core concept here is statistical arbitrage:

Statistical Analysis: ML algorithms, particularly those focused on statistical learning, analyze historical price relationships to determine the normal bounds of divergence between the pair. When the current market prices deviate significantly from this historical norm, it signals a potential trade.

Feature Engineering: Key features such as price ratios, price differences, historical volatility, and mean reversion metrics are created and used as inputs into ML models.

Classification and Regression: ML models like SVM (Support Vector Machine), Logistic Regression, or Random Forests are employed to predict whether the spread between pairs will converge or diverge in the future.

Deep Learning (DL) in Pair Trading

Deep Learning can process vast datasets with many nonlinear relationships more effectively than traditional ML techniques:

Pattern Recognition: DL models, particularly Neural Networks, are adept at identifying complex patterns in data, which can be crucial for recognizing trading signals that simpler models might miss.

Data Rich Environment: Given the high dimensionality in financial data, DL can discern subtle relationships between pairs that are not apparent to human traders or simpler algorithms.

Anomaly Detection: Deep learning can be used to detect anomalies in price movements between paired assets, suggesting opportunities for trades when anomalies occur outside of historical norms.

Reinforcement Learning (RL) in Pair Trading

Reinforcement Learning differs from supervised learning by learning to make decisions from actions based on trial and error, making it well-suited for dynamic and uncertain environments like financial markets:

Policy Learning: In pair trading, an RL agent can learn a policy to decide when to enter and exit trades by receiving rewards based on the profitability of its actions.

Simulation and Back-testing: RL algorithms can be continuously trained and tested within simulated environments to refine strategies before deployment in real markets.

Adaptation to Market Conditions: RL models can adapt to changing market conditions dynamically, learning from new data as it becomes available without the need for retraining the model from scratch.

Following a thorough review of the quantitative trading methodologies documented in the literature, I have systematically organized and categorized these methods into the tables presented below. Table 10 presents a compilation of the conventional supervised learning models that were employed, while Table 11 shows a set of unsupervised learning models that were used. Table 12 outlines the standard deep learning models utilized in the study. Table 13 details the commonly used reinforcement learning models, while Table 14 provides an overview of the hybrid models that were implemented. The primary methodologies outlined in the following tables will be elaborated upon in the subsequent sections of this chapter.

Table 10. Comprehensive overview of supervised learning models employed

Article	**Supervised Learning Models**
[2, 30, 47, 64, 67]	SVM

[62, 65, 66]	SVR
[4]	Elastic Net Regression (EN)
[16]	AdaBoost
[19]	XGBoost
[25, 39]	LGBM, RF, SVR
[54]	RF
[55]	LR, RF
[59]	GBDT
[60]	SVM, RF, ANFIS

The supervised learning models used in the reviewed studies include a diverse range of techniques, each with its own strengths and applications. Support Vector Machines (SVM) and Support Vector Regression (SVR) are both based on the principles of maximizing the margin between data points and a decision boundary. SVM is primarily used for classification tasks, while SVR adapts the SVM methodology for regression, predicting continuous values. Both are effective in high-dimensional spaces and are known for their robustness in handling outliers.

Random Forest (RF) and Gradient Boosting Decision Trees (GBDT), including its variants like XGBoost and LightGBM (LGBM), are ensemble methods that combine multiple decision trees to improve predictive accuracy and control overfitting. While RF operates by averaging the predictions of individual trees to reduce variance, GBDT and its variants sequentially build trees to correct the errors of the previous ones, focusing more on reducing bias and enhancing predictive performance. XGBoost and LGBM further optimize GBDT by improving computational speed and accuracy through techniques such as regularization and efficient handling of large datasets.

Elastic Net Regression (EN) and Logistic Regression (LR) are linear models often used for regression tasks. EN combines the properties of L1 (Lasso) and L2 (Ridge) regularizations, which helps in feature selection and handling multicollinearity in datasets. LR is commonly employed for binary classification problems, using a logistic function to model the probability of categorical outcomes.

AdaBoost is a boosting algorithm that combines multiple weak classifiers to create a strong classifier. It adjusts the weights of incorrectly classified instances, directing subsequent models to focus more on difficult cases. This iterative process helps improve overall accuracy, particularly in classification tasks.

Adaptive Neuro-Fuzzy Inference System (ANFIS), found in one of the studies, integrates neural networks and fuzzy logic principles, making it adept at handling non-linear and complex systems by modeling uncertain and imprecise information.

In summary, these models share the goal of enhancing predictive performance but differ in their approaches—ranging from linear and ensemble methods to those that integrate machine learning with fuzzy logic. Ensemble methods like RF, GBDT, and their variants generally offer higher accuracy and robustness compared to simpler models like LR and SVM, especially on large, complex datasets. However, they also require more computational resources. Understanding these similarities and differences is essential for selecting the appropriate model based on the specific requirements of the trading strategy and data characteristics.

Table 11. Comprehensive overview of unsupervised learning models employed

Article	Unsupervised Learning Models
[13, 49]	DBSCAN
[7]	PCA, DBSCAN
[9]	K-Means, DBSCAN, AHC
[18]	OPTICS Clustering Algorithm (OPTICS)
[68]	EM Algorithm (EM)
[25, 39]	LGBM, RF, SVR

The unsupervised learning models highlighted in Table 11 encompass a variety of clustering and dimensionality reduction techniques, each with distinct characteristics and applications. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular clustering method that identifies clusters based on density, making it effective at discovering clusters of arbitrary shapes and handling noise in the data. It is particularly well-suited for datasets with varying densities but may struggle with high-dimensional data or clusters of differing density.

K-Means is another widely used clustering algorithm that partitions the dataset into a predefined number of clusters by minimizing the variance within each cluster. K-Means is computationally efficient and straightforward but assumes that clusters are spherical and evenly sized, which may not be suitable for all datasets. Agglomerative Hierarchical Clustering (AHC) is a hierarchical method that builds clusters step-by-step by merging the closest pairs, forming a dendrogram that helps visualize the clustering process. While it provides a flexible approach to clustering, it can be computationally expensive on large datasets.

PCA (Principal Component Analysis) is a dimensionality reduction technique that transforms data into a lower-dimensional space by projecting it along the directions of maximum variance. Although not a clustering method, PCA is often used in combination with clustering algorithms like DBSCAN to preprocess data and reduce noise or redundancy.

OPTICS (Ordering Points To Identify the Clustering Structure) is a variant of DBSCAN that not only identifies clusters but also captures the hierarchical structure of the data. It is particularly useful when clusters vary significantly in density, as it does not require a predefined cluster count and adjusts to the data's intrinsic structure.

The EM Algorithm (Expectation-Maximization) is used for finding maximum likelihood estimates in models with latent variables, often applied in the Gaussian Mixture Model (GMM) clustering. It assumes that data points are generated from a mixture of several Gaussian distributions, making it flexible but sensitive to initial conditions and computationally intensive on large datasets.

Lastly, LGBM (LightGBM), RF (Random Forest), and SVR (Support Vector Regression) are primarily supervised learning models but are included here possibly due to their unsupervised adaptations or combined applications in certain studies.

In summary, these unsupervised learning models share the common goal of identifying patterns and structures in unlabeled data but differ in their assumptions, approaches, and suitability for various data types. DBSCAN and OPTICS excel in handling noise and irregularly shaped clusters, while K-Means and AHC provide more structured clustering options. PCA aids in data reduction and visualization, and the EM Algorithm offers a probabilistic clustering framework. Understanding these similarities and differences is crucial for selecting the appropriate model based on the specific characteristics of the dataset and the research objectives.

Table 12. Comprehensive overview of deep learning models employed

Article	Deep Learning Models
[24, 36, 37, 38, 48]	LSTM
[1]	Neural Network (NN)
[17]	KalmanNet Bollinger Trading (KalmanBOT)
[27]	Stochastic Neural Network (SNN)
[28]	RNN, LSTM, TCN
[31]	TCN
[45]	LSTM, LSTM Encoder-Decoder
[46]	LSTM, CNN, Multilayer perceptron (MLP)
[50]	DNN
[56]	BNN, DP-BNN, DDP-BNN
[57]	Filterbank CNN

The deep learning models employed in the reviewed studies encompass a diverse range of neural network architectures, each designed to tackle specific challenges in financial modeling and pairs trading. Long Short-Term Memory (LSTM) networks are recurrent neural networks (RNNs) particularly suited for time series analysis due to their ability to learn long-term dependencies and remember information over extended sequences. This makes LSTM ideal for modeling stock prices and financial time series, where the ability to retain past information is crucial.

Recurrent Neural Networks (RNNs), similar to LSTMs, are designed for sequence data but can suffer from short-term memory and vanishing gradient problems, which LSTMs address with their unique cell state and gating mechanisms. Temporal Convolutional Networks (TCNs) also handle sequence data but use convolutional layers with dilations to capture long-term dependencies, offering an alternative to LSTMs by providing better parallelism and more stable gradients.

Neural Networks (NNs) and Deep Neural Networks (DNNs) represent basic feedforward structures with varying depths, where DNNs involve multiple hidden layers to capture complex patterns in data. Multilayer Perceptrons (MLPs) are a type of feedforward neural network with one or more layers of neurons between input and output layers, commonly used for classification and regression tasks.

Convolutional Neural Networks (CNNs), typically used for image and grid-like data, are adapted here for time series through techniques like Filterbank CNNs, capturing spatial hierarchies and local patterns. This is beneficial for recognizing repetitive patterns and trends in stock price movements.

KalmanNet Bollinger Trading (KalmanBOT) combines deep learning with Kalman filtering, integrating state estimation with trading strategies, particularly effective for capturing and adjusting to dynamic market conditions.

Stochastic Neural Networks (SNNs) introduce randomness into the network's activations or weights, which can help the model explore multiple solutions and improve generalization, a useful feature in the unpredictable nature of financial markets.

Bayesian Neural Networks (BNNs) and their variations (DP-BNN, DDP-BNN) incorporate uncertainty in model predictions, offering a probabilistic approach that quantifies uncertainty in predictions, which is crucial in financial decision-making where risk management is paramount.

LSTM Encoder-Decoder models extend the LSTM architecture to sequence-to-sequence tasks, where an input sequence is encoded into a fixed-length context vector, which is then decoded into an output sequence. This is particularly useful for forecasting sequences of stock prices over time.

In summary, while all these models aim to leverage deep learning's strengths in pattern recognition and prediction, they differ in their handling of temporal dependencies, uncertainty, and the nature of data input. LSTMs, RNNs, and TCNs are suited for sequential data with varying strengths in capturing temporal patterns, CNNs excel in local pattern recognition, and models like BNNs and SNNs add robustness through probabilistic reasoning. Understanding these similarities and differences aids in selecting the appropriate model based on the specific characteristics of the financial data and the desired outcomes of the trading strategy.

Table 13. Comprehensive overview of reinforcement learning models employed

Article	**Reinforcement Learning Models**
[32, 53]	Deep Q-network (DQN)
[5, 33, 41]	DRL, CA-DRL
[3]	Two-Level Reinforcement Learning
[6]	CA-DRL, NEWS-CO-DRL
[10]	PPO, PPO-PT, PPO-PT w/o Demo, SAPT, PTDQN
[12]	Hierarchical Reinforcement Learning (HRL)
[15]	P-DDQN, PTDQN
[29]	SAPT, SAPT w/o Break, SAPT w/o Time, SAPT w/o Hold, PTDQN, SAPT-3-std, SAPT-ADF, SAPT-BCD, SAPT-LSTM
[34]	RL-0, RL-0.02, RL-0.05, RL-0.1
[35]	Deep Reinforcement Learning (DRL)
[42]	DQN, Double Deep Q-Network (DDQN)
[63]	RL

The reinforcement learning models reviewed in Table 13 include a diverse array of approaches, each tailored to specific decision-making and optimization tasks within financial markets. Deep Q-Networks (DQN) and their variants like Double Deep Q-Networks (DDQN) are widely used due to their ability to handle high-dimensional state spaces. DQNs utilize a neural network to approximate the Q-values, allowing agents to learn optimal policies through trial and error. DDQNs improve upon DQNs by addressing the overestimation bias in Q-value predictions, leading to more stable and accurate learning.

Deep Reinforcement Learning (DRL) encompasses methods that integrate deep learning with reinforcement learning, allowing for the processing of large amounts of data and the handling of complex environments. Models such as CA-DRL (Cointegration Approach-DRL) extend DRL specifically for pairs trading by incorporating cointegration principles, enhancing the model's ability to exploit long-term relationships between assets.

Proximal Policy Optimization (PPO) and its variations, including PPO-PT and versions without demonstrations, are policy gradient methods that optimize policies through iterative updates while ensuring that each update stays within a predefined threshold, improving stability and performance. These methods are particularly effective in continuous action spaces, making them suitable for fine-tuning trading strategies.

Hierarchical Reinforcement Learning (HRL) breaks down decision-making into multiple levels or hierarchies, allowing the agent to learn complex tasks by dividing them into simpler sub-tasks. This approach can lead to more efficient learning and better scalability when dealing with multi-step trading strategies or managing portfolios with diverse assets.

Two-Level Reinforcement Learning and other hierarchical methods, such as those used in SAPT (Structural Adjustment and Policy Testing), offer modular approaches where each level addresses a specific aspect of the trading decision, such as timing or holding periods. These models are particularly useful when the trading strategy involves sequential decision-making with multiple criteria.

P-DDQN (Prioritized Double Deep Q-Network) and PTDQN (Prioritized Twin Delayed Q-Network) incorporate prioritization in experience replay, giving priority to experiences that are more informative for the learning process. These techniques improve the learning efficiency and convergence speed by focusing on more critical transitions.

Two-Level Reinforcement Learning, SAPT, and related models introduce multiple reinforcement learning frameworks within the trading strategy, each tailored to a different component of the decision-making process, such as break detection or time and hold strategies. This layered approach allows the models to tackle complex trading environments with a more refined strategy that considers various market factors.

In summary, these reinforcement learning models share a common goal of optimizing trading decisions through adaptive learning and feedback from the market environment. They differ primarily in how they handle exploration vs. exploitation, manage action spaces, and address multi-step decision processes. By incorporating hierarchical structures, prioritization, and cointegration methods, these models are adapted to specific financial contexts, allowing for more targeted and effective trading strategies. Understanding these nuances is crucial for selecting the right model based on the complexity of the trading task and the characteristics of the financial data.

Table 14. Comprehensive overview of combined models employed

Article	Combined Models
[8]	PCA, Convolutional AutoEncoders (CAE), DBSCAN, PPO
[14]	LR, XGBoost, CNN, LSTM
[20]	A-LSTM+1/N, RF+MVF, RF+1/N, SVM+MVF, SVM+1/N
[22]	LSTM, OPTICS
[23]	XGBoost, TCAN
[26]	SVM, XGBoost, DNN, LSTM, RAF
[40]	LSTM, ANN, LR, GBDT, FeedForward NN
[44]	LSTM, CNN, Deterministic Policy Gradient (DPG)
[51]	ANN, XGBoost
[52]	LR, Gaussian Discriminant Analysis (GDA), SVM, NN
[42]	DQN, Double Deep Q-Network (DDQN)
[61]	DNN, GBDT, RF

5.1 Machine Learning

In the field of computer science, particularly within the subfield of machine learning, machines are trained to learn from data programmatically. Machine learning algorithms can be broadly categorized into two main types: supervised learning and unsupervised learning. To train a machine learning model, it is necessary to supply both the algorithm and the relevant training data, which the model uses to adjust its parameters. Various machine learning models have been applied to predict stock market movements. The following sections will detail the methods previously summarized in the literature, with a specific focus on the techniques outlined in Tables 10, 11 and 14.

5.1.1 Supervised Learning

Supervised learning involves training a machine learning algorithm using input data that is paired with corresponding labels. The goal is for the model to approximate the function $y = f(x)$ as accurately as possible. This approach utilizes a labeled training dataset to guide the algorithm. Supervised learning algorithms are categorized into:

(a) Regression: This type is defined by the nature of the output variable. When the output variable is continuous, the task is identified as a regression problem. Examples of regression tasks include predicting the price of a house or forecasting stock prices.

(b) Classification: Classification tasks involve categorical output variables, such as color or shape. Most machine learning applications that deal with discrete output labels utilize supervised learning approaches. Common supervised learning techniques include logistic regression, linear regression, Support Vector Machine (SVM), and random forests.

From the analysis presented in Tables 10 and 14, it is evident that Support Vector Machines (SVM) and Random Forests (RF) are the predominant methods employed in the surveyed literature. Specifically, SVM was applied in nine papers, corresponding to a utilization frequency of 13%, while RF was adopted in six articles, achieving a usage rate of 9%. This analysis underscores the significant reliance on these algorithms within the research community. The subsequent sections will delve deeper into the application of SVM within the context of pair trading, examining their efficacy and the specific roles they play in enhancing trading strategies.

5.1.1.1 Introduction to Support Vector Machine (SVM)

Support Vector Machine is a powerful and versatile machine learning algorithm used for both classification and regression tasks, though it is more commonly used for classification. It belongs to the category of supervised learning, where the model is trained with data that includes the input features and the corresponding output labels. Here is a breakdown of how SVM works.

Firstly, it is imperative to maximize the margin. SVM operates on the principle of finding a hyperplane that best separates the classes in the feature space. The objective is to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. These nearest points are called support vectors, as they support the hyperplane.

Then linear and non-linear classification is needed. In cases where the data is linearly separable, SVM finds a straight line (in two dimensions), a plane (in three dimensions), or a hyperplane (in higher dimensions) that separates the classes. When the data is not linearly separable, SVM uses a method called the kernel trick to transform the input space into a higher dimensional space where a hyperplane can be used for separation.

The kernel trick involves using a kernel function to compute the dot product of the transformed data, enabling operations in a high-dimensional space without explicitly transforming all data points. Common kernels include polynomial, radial basis function (RBF), and sigmoid.

The last step is regularization. SVM includes a regularization parameter, often denoted as C, which controls the trade-off between achieving a low error on the training data and minimizing the model complexity for better generalization. A higher value of C tries to classify all training examples correctly, whereas a lower value of C allows more slack for misclassification but aims for a simpler decision boundary that may generalize better.

Mathematically, the hyperplane can be represented by the equation:

$$W^T \cdot \mathrm{X} + \mathrm{b} = 0 \tag{5.1}$$

Where, W is the normal vector to the hyperplane, determining the direction of the hyperplane. X represents the data points in the feature space. b is the bias term, which adjusts the position of the hyperplane along the normal vector.

In SVM, the goal is to choose W and b such that the margin, the minimum distance between the nearest data points (support vectors) and the hyperplane, is maximized. For linearly separable cases, training an SVM involves solving the optimization problem.

The optimization process prioritizes the maximization of the margin, which is achieved by identifying a hyperplane that maximizes the geometric margin between the closest points of differing classes. This process is mathematically represented as the minimization of $\frac{1}{2}||\mathrm{W}||^2$, offering a simplified solution. Furthermore, the model is constrained such that for every training data point $(X_i,\ y_i)$, the following condition holds: $y_i\ (W^T \cdot \mathrm{X} + \mathrm{b}) \geq 1$, where, y_i signifies the class label, typically designated as either +1 or -1. In scenarios involving non-linearly separable data, the kernel trick facilitates the transformation of the original feature space to a higher-dimensional one, allowing the separation with a hyperplane. Employing kernels like the polynomial and radial basis function (RBF), the decision function is formulated as

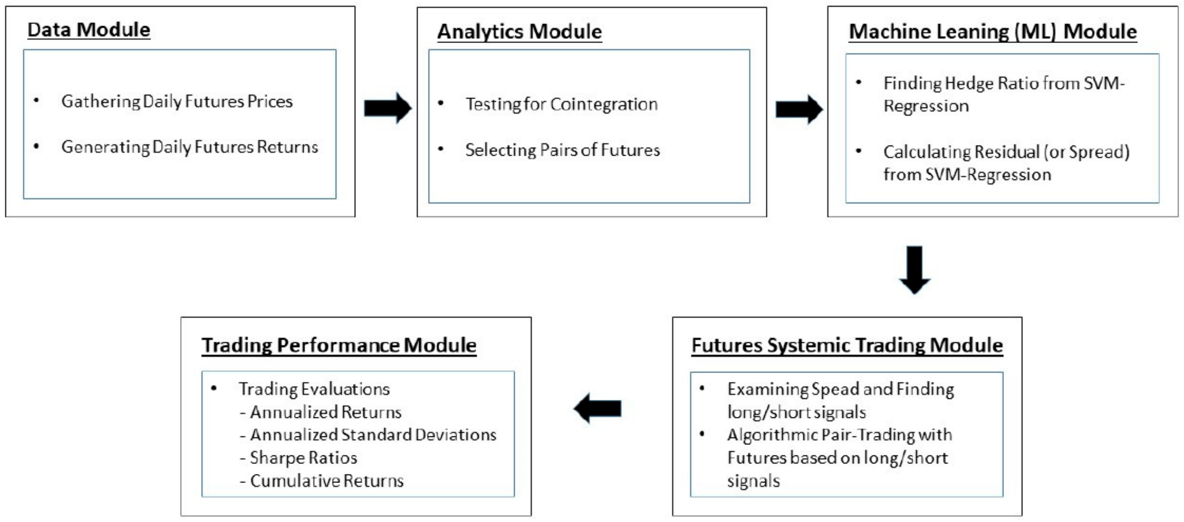
$$f(X) = sgn(\sum_{i=1}^{n} \alpha_i y_i K(X_i,\ X) + b) \tag{5.2}$$

where, α_i are the Lagrange multipliers derived during training via the dual problem, $K(X_i,\ X)$ is the selected kernel function computing the inner products in the expanded feature space, and sgn denotes the sign function, imperative for the classification decision.

5.1.1.2 Support Vector Machine (SVM) in Pair Trading

Speaking to the steps of implementing Support Vector Machine (SVM) in pair trading, as shown in Figure 4 from article [47], the model consists of five modules, including a data step procedure, analytics procedures, hedging based on the machine learning method, pairs trading, and evaluation of trading performance.

Figure 4: Systematic pair trading model architecture using SVM for futures in article [47]


Data Module
• Gathering Daily Futures Prices
• Generating Daily Futures Returns
Analytics Module
• Testing for Cointegration
• Selecting Pairs of Futures
Machine Leaning (ML) Module
• Finding Hedge Ratio from SVM-Regression
• Calculating Residual (or Spread) from SVM-Regression
Futures Systemic Trading Module
• Examining Spead and Finding long/short signals
• Algorithmic Pair-Trading with Futures based on long/short signals
Trading Performance Module
• Trading Evaluations
- Annualized Returns
- Annualized Standard Deviations
- Sharpe Ratios
- Cumulative Returns

The specific steps for applying SVM to Pair trading are also given in the article [30]. Here is a summarized sequence of steps for implementing SVM in pair trading, as outlined in article [30]. Model performance is also given in **Table 15**.

Table 15. Model performance with different parameters in article [30]

Pair	Accuracy	AUC of ROC	Parameters
600797.SH-603421.SH	64%	0.63	kernel=poly, C=0.1, degree=3, coef0=2
002807.SZ-601288.SH	57%	0.57	kernel=poly, C=1, degree=3, coef0=0.5
000025.SZ-000715.SZ	56%	0.57	kernel=rbf, C=0.1, gamma=0.01, coef0=0
600029.SH-600115.SH	58%	0.59	kernel=poly, C=4, degree=3, coef0=5
002696.SZ-600975.SH	66%	0.67	Kernel=poly, C=2, degree=2, coef0=2

Data Acquisition and Preprocessing

Data Sources and Retrieval: Utilize the Tushare API to fetch historical price data for a predetermined list of stock pairs from the Chinese market. This data includes open, high, low, close, and volume metrics for each trading day.

Data Cleansing: Handle missing values, correct anomalies, and remove non-trading days from the dataset to ensure data quality.

Normalization: Apply Min-Max normalization to scale the features to a uniform range (typically 0 to 1), enhancing the convergence speed and stability of the SVM algorithm during training.

Feature Engineering and Selection

Price Spread: Calculate the absolute and relative price differences between the two stocks in each pair.

Volatility Measures: Compute rolling standard deviations of price spreads to capture the volatility dynamics between the pairs.

Technical Indicators: Generate features like moving averages, RSI, and MACD to incorporate momentum and trend-following aspects into the model.

Feature Reduction: Employ techniques such as Principal Component Analysis (PCA) or feature importance metrics from preliminary tree-based models to reduce the dimensionality of the feature space, focusing on the most informative predictors.

SVM Configuration and Model Training

Kernel Choice and Parameterization: Evaluate different SVM kernels (linear, polynomial, and RBF) on a subset of the data using cross-validation to determine the most effective kernel based on prediction accuracy and computational efficiency.

Cross-Validation and Grid Search: Implement a grid search over a range of values for C (penalty parameter) and gamma (kernel coefficient for RBF), along with polynomial degree where applicable, to find the optimal parameter set that minimizes overfitting and maximizes out-of-sample prediction accuracy.

Model Training: Train the SVM model on the full training dataset using the selected parameters and kernel, ensuring the model captures complex patterns and interactions in the data.

Strategy Formulation and Trading Signal Generation

Signal Derivation: Use the trained SVM to predict changes in the price spread—whether it is likely to widen or narrow.

Trade Execution Criteria: Define specific criteria for trade execution based on the SVM output, such as entering a long/short position on the pair when the predicted spread widening/narrowing exceeds a certain threshold.

Back-testing Strategy: Simulate the trading strategy using historical data to validate the effectiveness of the SVM predictions and the predefined trading criteria.

Back-testing and Performance Evaluation

Historical Simulation: Conduct rigorous back-testing by applying the SVM-based strategy over several historical periods to assess consistency and resilience across different market conditions.

Performance Metrics: Calculate various metrics such as cumulative returns, Sharpe ratio, maximum drawdown, and beta-adjusted performance to evaluate the strategy's risk-adjusted returns compared to a benchmark.

Implementation and Real-Time Execution

Real-Time Data Feeding: Implement a system to fetch and process real-time data that feeds into the SVM model continuously.

Trade Execution: Automate the trade execution process based on real-time predictions and predefined trading signals, integrating with brokerage APIs to place trades automatically.

Risk Management: Implement dynamic risk management protocols, including stop-loss, take-profit orders, and position sizing based on the volatility of the stock pairs and overall market conditions.

Ongoing Monitoring and Model Updating

Performance Monitoring: Continuously monitor the performance of the trading strategy, tracking live results against historical back-tests.

Model Recalibration: Regularly recalibrate and retrain the SVM model using the latest data to adapt to new market dynamics, ensuring that the model remains predictive and relevant.

Feedback Loops: Integrate feedback mechanisms to adjust the trading strategy based on performance outcomes and market feedback, enhancing the adaptability of the approach.

5.1.2 Unsupervised Learning

Unsupervised learning is a type of machine learning where the algorithm is trained using input data without any corresponding output labels. The primary objective of unsupervised learning is to model the underlying structure or distribution in the data to learn more about the data itself. This approach is used to discover patterns and relationships in datasets where the outcomes or labels are not known ahead of time.

Unsupervised learning can be divided into two main tasks: Clustering and Association. Clustering algorithms, such as k-means, aim to identify inherent groups or clusters within the data, organizing unlabeled data points into meaningful subgroups. This is useful for segmenting data into distinct categories based on similarities among the data points. On the other hand, association algorithms identify sets of items or events that often occur together in the dataset, which is helpful in market basket analysis and recommendation systems. Through these methods, unsupervised learning provides valuable insights from data where the information about the output is absent.

The data outlined in Tables 11 and 14 clearly demonstrates a preference for Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Principal Component Analysis (PCA) within the scope of the reviewed academic works. In particular, DBSCAN has been implemented in four papers, making up 6% of the applications, while PCA has been utilized in two papers, constituting a 9% application share. These findings highlight the prominent adoption of these methodologies among researchers. The forthcoming sections aim to explore the integration of DBSCAN in the sphere of pair trading, critically evaluating their effectiveness and the distinct contributions they make to the refinement of trading strategies.

5.1.2.1 Introduction to Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Developed in 1996 by Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a renowned clustering algorithm that groups points based on the density of their neighborhoods. It distinguishes between points that are densely packed and those in sparse areas, identifying the latter as outliers. This non-parametric, density-based method does not require prior knowledge of the number of clusters, making it widely utilized and frequently referenced in clustering tasks.

The DBSCAN algorithm operates on two primary inputs: ε (epsilon), which is the radius that determines a point's neighborhood, and $minPts$, which is the threshold that defines the minimum number of points required to establish a dense region. The process initiates with a randomly selected unvisited point. The ε-neighborhood of this point is examined; if it meets or exceeds the $minPts$ threshold, it becomes the seed of a new cluster. If not, the point is provisionally classified as noise, although this classification may change if the point falls into the ε-neighborhood of a different point that meets the density requirement.

When a point is identified as a dense component of a cluster, the points within its ε-neighborhood are also included in the cluster. Subsequently, their own ε-neighborhoods are analyzed for density and added to the cluster if they fulfill the criteria. This iterative process of expansion continues until the cluster is fully delineated. Thereafter, the algorithm selects another unvisited point and repeats the process, facilitating the formation of additional clusters or the identification of noise.

Consider a set of points in some space to be clustered. Let ε be a parameter specifying the radius of a neighborhood with respect to some point. For the purpose of DBSCAN clustering, the points are classified as core points, reachable points and outliers, as follows:

A point p is a core point if at least $minPts$ points are within distance ε of it (including p).

A point q is directly reachable from p if point q is within distance ε from core point p. Points are only said to be directly reachable from core points.

A point q is reachable from p if there is a path $p_1, \dots, p_n$ with $p_1 = p$ and $p_n = q$, where each $p_i + 1$ is directly reachable from p_i. Note that this implies that the initial point and all points on the path must be core points, with the possible exception of q.

All points not reachable from any other point are outliers or noise points.

Now if p is a core point, then it forms a cluster together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.

The methodology of DBSCAN can be encapsulated in the following procedural steps:

Assess the ε-neighborhood of every point in the dataset to pinpoint the set of core points that have at least $minPts$ within their vicinity.

Construct the connectivity graph of core points, disregarding points that do not meet the core criteria, and subsequently identify the clusters through this graph.

Integrate each peripheral non-core point into a corresponding cluster if it is within an ε distance from the cluster, or else categorize it as noise.

5.1.2.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN) in Pair Trading

In the context of pair trading, DBSCAN can be effectively applied to identify pairs of stocks that move in a similar fashion over time. DBSCAN can be used in the preliminary stages of a pair trading strategy to identify clusters of stocks that exhibit similar price movements. By grouping stocks into clusters based on their historical price data, traders can more efficiently identify potential pairs for trading.

The paper [9] presents a pairs trading strategy with DBSCAN, going beyond traditional methods that rely solely on price data by incorporating firm characteristics. This novel approach enhances strategy performance, as demonstrated within the US stock market over a 40-year period, resulting in notable annualized returns and Sharpe ratios. The strategy persists as profitable even when transaction costs are accounted for and is robust against data-snooping concerns.

Here is a detailed look at how DBSCAN can be utilized in pair trading by combining with the paper [9].

How DBSCAN Works

Feature Selection: Select features based on stock price data, such as returns over various time frames or other statistical measures like volatility or beta. As in the paper [9], clustering stocks based on past returns and firm characteristics to form groups of stocks with historical co-movement and similar features, employing a variety of characteristics that are forward-looking to decrease the likelihood of identifying spurious pairings.

Distance Metric: Define a suitable distance metric (e.g., Euclidean, Manhattan) to measure the similarity between the stocks based on the selected features.

Clustering: Apply DBSCAN to the feature set. Addressing the challenges of cluster size imbalance, where DBSCAN tends to produce a large cluster containing most stocks and identifies the remainder as outliers, indicating a high-density cluster within the high-dimensional data structure. This requires adjustment of clustering parameters and consideration of the number of clusters' impact on strategy performance. This is particularly useful in finance to distinguish between common movements and anomalies.

Cluster Analysis: Analyze the clusters to identify potential pairs. Stocks within the same cluster should have similar price movement patterns and can be considered for pair trading. As in the paper [9], formulating trading rules based on the divergence of past one-month returns within each cluster. Stocks with low (high) returns are considered undervalued (overvalued), adopting a contrarian strategy where undervalued stocks are bought and overvalued stocks are sold short. Positions are opened when the return difference exceeds one cross-sectional standard deviation from the past month's return differences and are rebalanced monthly. The result is shown in **Table 16**.

Table 16. Annualized risk-return metrics with DBSCAN in article [9]

DBSCAN	Long	Short	Long-Short
Mean return	0.291	0.053	0.238
Standard deviation	0.244	0.188	0.152
Sharpe ratio	1.195	0.281	1.571
t-statistic	2.296	2.215	2.875
Downside deviation	0.171	0.162	0.066
Sortino ratio	1.703	0.327	3.591
Gross profit	17.544	9.051	12.733
Gross loss	-5.603	-6.882	-2.96
Profit factor	3.131	1.315	4.302
Profitable years	37	27	38
Unprofitable years	4	14	3
Maximum drawdown	-0.477	-0.624	-0.129
Calmar ratio	0.611	0.085	1.846
Turnover	0.932	0.981	1.913

Also, article [9] conducted robustness tests, in **Table 17**, to assess the sensitivity to clustering parameters and to ensure results are not driven by data snooping.

Table 17. Parameter sensitivity of the strategies, DBSCAN with different percentiles (α) for the outlier threshold in article [9]

α	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Mean return	0.256	0.2	0.173	0.154	0.148	0.148	0.138	0.134	0.129
Sharpe ratio	2.039	1.763	1.526	1.328	1.19	1.142	1.036	0.954	0.867
Maximum drawdown	-0.149	-0.146	-0.163	-0.206	-0.197	-0.242	-0.238	-0.285	-0.332
Number of clusters	2	2	2	2	2	2	2	2	1
Number of stocks in clusters	376(12.05)	749(24.06)	1092(35.02)	1418(45.44)	1729(55.41)	2034(65.14)	2328(74.56)	2609(83.54)	2878(92.2)
Number of outliers	2781(87.95)	2408(75.94)	2065(64.98)	1739(54.56)	1428(44.59)	1123(34.86)	829(25.44)	548(16.46)	279(7.8)
Number of stocks traded	75(2.47)	188(6.17)	314(10.18)	450(14.52)	594(19.09)	743(23.82)	896(28.67)	1053(33.67)	1217(38.86)

Practical Implementation Summarize

Data Preparation: Collect and preprocess historical price data for a basket of stocks.

Parameter Selection: Tune the two main parameters of DBSCAN – epsilon (the maximum distance between two samples for one to be considered as in the neighborhood of the other) and min_samples (the number of samples in a neighborhood for a point to be considered as a core point).

Execution and Validation: Apply DBSCAN to identify clusters, and then examine these clusters to choose potential pairs. Validate these pairs through back-testing to assess the feasibility and profitability of the trading strategy.

Benefits of Using DBSCAN in Pair Trading

Efficiency: Reduces the dimensionality of the problem by filtering out stocks that do not exhibit strong similarities, thus narrowing the focus to only promising pairs.

Outlier Detection: Automatically identifies and excludes outlier stocks that could potentially skew the performance of a pair trading strategy.

Adaptability: DBSCAN does not require a pre-specified number of clusters, making it adaptable to various market conditions and datasets.

Challenges and Considerations

Parameter Sensitivity: Choosing the right parameters for epsilon and min_samples are crucial and can significantly affect the outcome. These parameters might need to be adapted over time as market conditions change.

Market Noise: Financial markets are inherently noisy and influenced by numerous external factors, which can affect the stability and reliability of the clusters identified by DBSCAN.

5.2 Deep Learning

Deep learning refers to a subset of machine learning techniques that employ artificial neural networks (ANNs) characterized by multiple layers, which is signified by the term "deep" in its name. These layers enable the model to engage in representation learning, enhancing its ability to identify and utilize abstract patterns in data. The architectures utilized in deep learning include a variety of networks such as deep neural networks, deep belief networks, recurrent neural networks, convolutional neural networks, and transformers. These models are versatile and can be trained through supervised, semi-supervised, or unsupervised learning methods.

The application of deep learning spans a broad range of fields. In computer vision and speech recognition, it helps machines interpret and understand visuals and spoken words, respectively. It also plays a pivotal role in natural language processing and machine translation, enabling better communication and understanding between humans and machines. Additionally, deep learning is used in bioinformatics and drug design to predict molecular activities and interactions, while in medical image analysis, it aids in diagnosing diseases from images more accurately. The technology also extends to climate science for modeling and prediction, material inspection for quality control, and even to mastering complex board games, often achieving performance levels that meet or exceed those of human experts.

Quantitative trading is one of the applications of deep learning. In the forthcoming sections, I will thoroughly explore how deep learning is applied to pair trading. The discussion will expand on the methodologies briefly mentioned earlier in the literature, with a particular emphasis on the techniques specified in Tables 12 and 14.

The data derived from Tables 12 and 14 clearly highlights that Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) are the most commonly used techniques in the reviewed literature. Specifically, LSTM was utilized in eleven studies, accounting for a usage frequency of 16%, whereas CNN was implemented in four papers, representing a usage rate of

6%. This evidence points to a substantial reliance on these algorithms within the scholarly community. The following sections will further investigate the use of LSTM in the realm of pair trading, assessing their effectiveness and detailing the roles they fulfill in improving trading strategies.

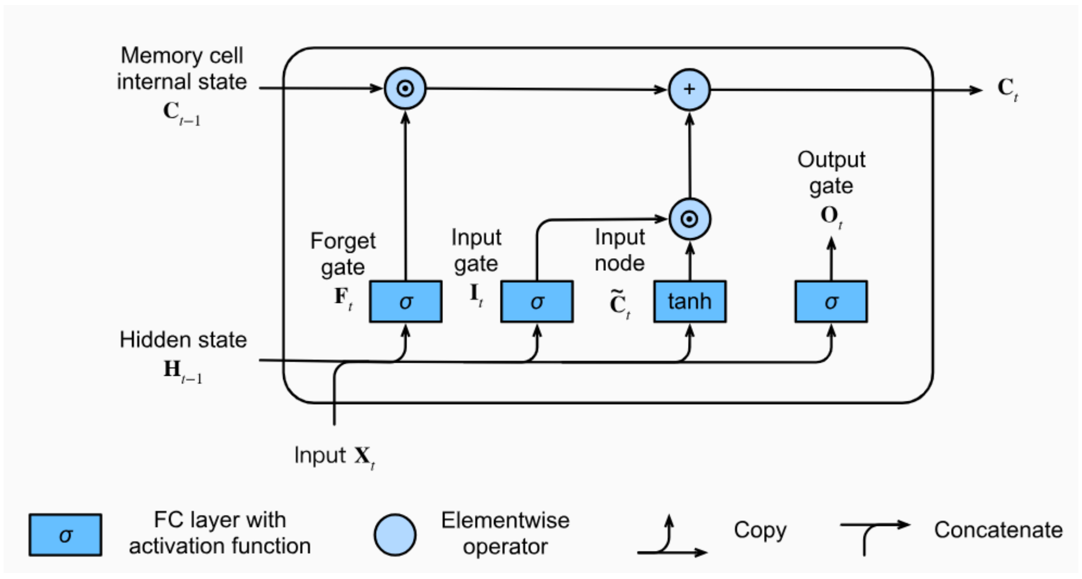
5.2.1 Introduction to Long Short-Term Memory (LSTM)

A recurrent neural network (RNN) is a type of neural network designed to process sequential data. Unlike standard neural networks, RNNs can process data that changes over time. For example, the meaning of a word may change based on the context provided by the previous content, and RNNs are good at solving such challenges.

Long Short-Term Memory (LSTM) is a special type of RNN, primarily developed to address issues of vanishing and exploding gradients that can occur during the training of long sequences. Simply put, compared to ordinary RNNs, LSTMs perform better over longer sequences.

What sets LSTMs apart from traditional neural networks is their unique memory cells, which allow them to maintain information in 'memory' for long periods of time. An LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. These components work together to regulate the flow of information into and out of the cell, as well as to manage the cell's state over time. Here are how these gates function:

Figure 5. Schematic Representation of the LSTM Model


Memory cell internal state $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
Output gate $\mathbf{O}_t$
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Input node $\tilde{\mathbf{C}}_t$
tanh
σ
Hidden state $\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

Source: Zhao, W. and Fan, L. (2024). Short-Term Load Forecasting Method for Industrial Buildings Based on Signal Decomposition and Composite Prediction Model

Cell State (C_t):

This is the key component that sets LSTMs apart. The cell state is somewhat like a conveyor belt running through the entire chain of LSTM units. It has the ability to carry relevant information throughout the processing of the sequence. This means that information can be untouched or modified through the operations of gates, thus carrying key information through the sequence of data.

Forget Gate (f_t):

The forget gate decides what information should be thrown away or kept. Input from the previous hidden state (h_{t-1}) and the current input (x_t) are passed through a sigmoid function. Values come out between 0 and 1, where 0 means "completely forget this" and 1 means "completely keep this."

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.3}$$

Here, σ is the sigmoid function that outputs values between 0 and 1. W_f is the weight matrix for the forget gate, b_f is the bias term, h_{t-1} is the hidden state from the previous time step, and x_t is the current input vector. The output, f_t , is element-wise multiplied with the previous cell state C_{t-1} to determine which values are retained.

Input Gate (i_t) and Candidate Values ($\tilde{C}_t$):

The input gate decides the values that will be updated, and a $tanh$ layer creates a vector of new candidate values that could be added to the state. This is combined in a way to update the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.4}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5.5}$$

The input gate i_t uses a sigmoid function to decide which values of $\tilde{C}_t$ to accept. $\tilde{C}_t$, formed by a $tanh$ layer, represents candidate values that could be added to the cell state, thus enabling flexibility in updating state with non-linear transformations.

Update of Cell State (C_t):

The old cell state (C_{t-1}) is updated to the new cell state (C_t). The forget gate's output multiplies the old state (deciding what to forget), and the input gate's output multiplies the candidate values (deciding what to update).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.6}$$

This equation combines the past cell state C_{t-1} (modulated by f_t) with the new candidate values $\tilde{C}_t$ (modulated by i_t). This selective update allows LSTMs to carry forward relevant information through long sequences without the risk of exponential decay or growth of gradients.

Output Gate (o_t) and Hidden State (h_t):

Finally, the output gate decides what the next hidden state should be. The hidden state contains information on previous inputs. The hidden state is also used for predictions. A sigmoid layer decides which parts of the cell state make it to the output, and then we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.7}$$

$$h_t = o_t * tanh(C_t) \tag{5.8}$$

The output gate o_t determines which parts of the cell state C_t to send to the next hidden state h_t and potentially to the output layer of the network. This step involves filtering the cell state through a $tanh$ function to regulate the output values, ensuring non-linearity in the data representation.

The detailed dynamics and operational flow are as follows.

Sequential Dependency Management: Each component of the LSTM gates is specifically designed to handle different aspects of information control—retaining valuable historical information, updating it with relevant new data, and discarding redundancies. This design allows LSTMs to perform effectively across a wide range of sequential data lengths.

Gradient Flow Control: A critical advantage of LSTMs is their ability to control the flow of gradients during backpropagation. By using gates to regulate updates, LSTMs prevent gradients from vanishing (getting too small to make significant updates) or exploding (getting too large, leading to unstable updates). This is achieved by the gating mechanism, which selectively allows only certain gradients to pass through each timestep, depending on their relevance.

Learning Long-Term Dependencies: The unique structure of the LSTM allows it to learn which data is important to remember and which can be forgotten. This is particularly useful in tasks where context from the distant past influences the current decision, such as in language modeling where previous words provide context for the next word predictions.

By managing data flow through sophisticated gating mechanisms and maintaining a dynamic memory of important information, LSTMs provide a robust framework for tackling complex problems involving sequential data where context and history significantly impact current and future data interpretations.

5.2.2 Long Short-Term Memory (LSTM) in Pair Trading

Using Long Short-Term Memory (LSTM) networks in pairs trading involves leveraging their ability to remember and utilize historical financial data over extended periods, which is crucial for identifying and exploiting market inefficiencies between pairs of stocks, commodities, or other financial instruments.

The paper [38] provides a detailed methodology for using LSTM networks to identify and capitalize on pairs-trading opportunities. Here are the specific steps based on the information from the article.

Data Collection

Stock Selection: Focus on stocks within the S&P 500 index, which are collected annually to reflect changes in the index composition.

Historical Data: Gather daily price data (such as open, high, low, close, and volume) for a period spanning from January 2000 to June 2019. Use APIs like 'yfinance' to download daily historical data for selected stocks, focusing on adjusted close prices, which account for dividends and splits.

Data Preprocessing

Normalization: Apply Z-score normalization where the mean and standard deviation of each stock's price series are used to standardize the data.

Cointegration Testing: Use statistical software (e.g., R or Python with 'statsmodels') to perform Engle-Granger and Johansen tests to identify pairs of stocks whose price series exhibit statistically significant cointegration, indicating potential pairs trading opportunities.

LSTM Model Development

Input Features: The paper specifies using price gaps, trading volumes, and returns as input features. Each feature contributes differently to the model's prediction capability:

- Price Gaps: Calculated as the difference between a stock's price and the mean price of its cointegration group.
- Trading Volumes: Included to account for liquidity variations which might affect the price gap movements.
- Returns: Historical returns to capture the trend and volatility.

LSTM Configuration: The network likely uses several layers to adequately capture the temporal dependencies inherent in financial time series. While the exact number of layers and nodes per layer is not specified, a typical setup might include:

- Multiple LSTM layers to enhance learning depth.
- Dropout layers between LSTM layers to mitigate overfitting.
- A dense layer followed by a SoftMax activation to output probabilities for potential trading actions.

Model Training

Sequence Preparation: Create input sequences of fixed lengths (240 days as mentioned) that provide a window of historical data to the network.

Training Process: Train the model using backpropagation through time (BPTT), a common method for training LSTMs. Regularly validate the model on a held-out subset of the data to tune hyperparameters and prevent overfitting.

Optimization and Loss: Utilize optimizers like RMSprop or Adam, which are effective for stochastic gradient descent in time-series models, and use categorical cross-entropy as the loss function for the classification tasks.

Prediction

Execution of Prediction: After training, use the model to make predictions on new data. This involves feeding in new input sequences corresponding to the most recent data and obtaining predictions from the output layer.

Interpretation of Results: Convert the SoftMax probabilities into actionable trading signals. For example, a high probability of the price gap widening might trigger a buy signal for the underperforming stock and a sell signal for the overperforming one.

Strategy Implementation

Portfolio Construction: Based on the LSTM outputs, construct a trading strategy that involves buying stocks predicted to increase in price relative to their peers and selling those expected to decrease.

Risk and Money Management: Implement controls such as stop-loss orders, position sizing based on the confidence level of predictions, and maximum exposure limits to manage risk effectively.

Back-testing and Optimization

Back-testing Framework: Rigorously back-test the strategy using historical data to simulate trading performance. This involves running the LSTM model on historical data and applying the trading signals generated to see the hypothetical historical performance.

Parameter Tuning: Refine model parameters based on back-testing results. This could include adjusting the lookback period, the thresholds for trading signals, and the LSTM architecture.

Execution

System Integration: Implement the strategy in a live trading environment. This requires integration with market data feeds, execution platforms, and brokerage APIs.

Operational Monitoring: Monitor the system's performance in real-time, ensuring that execution occurs as planned without technical glitches.

Review and Adjustment

Continuous Improvement: Regularly review the strategy's performance and underlying model assumptions. Update the model with new data and retrain if necessary to adapt to new market conditions.

Strategic Adjustments: Based on performance reviews, make strategic adjustments to the trading algorithm, risk management tactics, and overall investment approach.

5.3 Reinforcement Learning

Reinforcement learning (RL) is a crucial domain at the intersection of machine learning and control theory, focusing on the strategies an intelligent agent should adopt within a dynamic environment to optimize the accumulation of rewards. As one of the fundamental machine learning paradigms—alongside supervised and unsupervised learning—RL is distinct in its approach.

Unlike supervised learning, RL does not rely on pre-labeled input/output pairs, nor does it require direct corrections for suboptimal actions. Instead, it emphasizes a strategic balance between exploring new possibilities and exploiting known strategies to maximize overall long-term benefits. Feedback in RL can be indirect or delayed, adding complexity to the learning process.

The conceptual framework for many RL environments is defined by a Markov decision process (MDP). While traditional dynamic programming methods used in RL presuppose complete knowledge of the MDP, modern RL algorithms operate effectively without a full mathematical description of the environment. This adaptability makes them suitable for large-scale MDPs where classical methods are impractical.

Quantitative trading stands out as a notable application of reinforcement learning. In the subsequent sections, this document will delve deeply into the use of reinforcement learning in pairs trading. The discussion will build upon methodologies previously referenced in the literature, with a focused analysis on the techniques detailed in Tables 13 and 14.

The information extracted from Tables 13 and 14 clearly demonstrates that Deep Q-network (DQN) and Deep Reinforcement Learning (DRL) are the predominant techniques in the literature surveyed. Specifically, DQN was employed in six studies, making up 9% of the instances, while DRL appeared in five papers, corresponding to a 7% usage rate. This data suggests a significant adoption of these algorithms within the academic field. Subsequent sections will further explore the application of DQN in pair trading, evaluating its efficacy and elaborating on its contribution to enhancing trading strategies.

5.3.1 Introduction to Deep Q-network (DQN)

Reinforcement learning involves an agent, a set of states S, and a set A of actions per state. By performing an action $a \in A$, the agent transitions from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score).

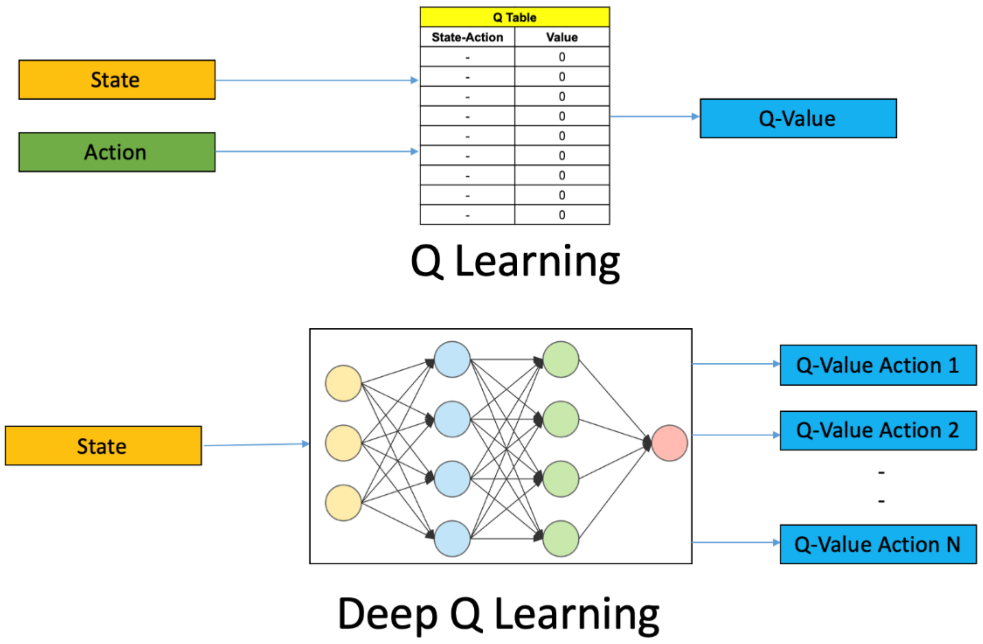
The goal of the agent is to maximize its total reward. It does this by adding the maximum reward attainable from future states to the reward for achieving its current state, effectively influencing the current action by the potential future reward. This potential reward is a weighted sum of expected values of the rewards of all future steps starting from the current state.

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.

For any finite Markov decision process, Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state. Q-learning can identify an optimal action-selection policy for any given finite Markov decision process, given infinite exploration time and a partly random policy. "Q" refers to the function that the algorithm computes – the expected rewards for an action taken in a given state.

Q-Learning worked well with small state space environments. The problem is that Q-Learning is a tabular method. This becomes a problem if the states and actions spaces are not small enough to be represented efficiently by arrays and tables. For the state spaces are gigantic, creating and updating a Q-table for that environment would not be efficient. In this case, the best idea is to approximate the Q-values using a parametrized Q-function $Q_{(s,\ a)}$.

Figure 6. Schematic Representation of Q-learning and Deep Q-learning



Source: A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python

Therefore, Deep Q-learning, as an extension of traditional Q-learning integrated with deep neural networks, has revolutionized the approach to handling environments with high-dimensional observation spaces. This method effectively employs the representational power of deep neural networks to learn optimal policies directly from raw sensory data, which is otherwise unmanageable for classical Q-learning due to its reliance on a discrete state-action space representation. Here’s a more detailed exploration of the fundamental components and the operation of Deep Q-learning:

Q-learning Review

At its core, Q-learning aims to learn the action-value function $Q(s,a)$ that indicates the expected utility of taking an action a in a state s and following the optimal policy thereafter. The goal is to maximize the sum of rewards, discounted over time, by updating the Q-values based on the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{5.9}$$

Here, α is the learning rate, γ is the discount factor, r is the reward, s' is the next state, and a' is a possible action in state s'.

Deep Neural Network as a Function Approximator

In Deep Q-learning, a deep neural network, often referred to as the Q-network, approximates the Q-function. The network takes the state s as input and outputs a vector of action values $Q(s,\cdot)$ for all possible actions. This approach overcomes the curse of dimensionality by not requiring a discrete table for Q-values and instead learning a continuous function that generalizes across similar states.

Experience Replay

To break harmful correlations between consecutive learning updates, Deep Q-learning leverages an experience replay mechanism. This process involves storing transitions (s,a,r,s') in a replay buffer and randomly sampling mini-batches from this buffer to train the network. This stochastic sampling method helps to stabilize the training process by providing a diverse range of experiences in each batch, mimicking independent and identically distributed data.

Fixed Q-targets

A critical issue in training neural networks in a reinforcement learning context is the moving target problem, where the constantly changing Q-values can lead to oscillations or divergence of the policy. Deep Q-learning addresses this by employing a separate network to generate the target Q-values. This target network has the same architecture as the Q-network but its weights are updated less frequently (e.g., every few thousand steps), thereby providing a stable target for the Q-network updates.

Loss Function and Optimization

The loss function used in Deep Q-learning is typically the mean squared error between the predicted Q-values and the Q-targets:

$$L(\theta) = E_{(s,a,r,s')\sim replay\,buffer}\,[(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta))^2] \tag{5.10}$$

where θ are the parameters of the Q-network and θ^{-} are the parameters of the target network. This loss is minimized using standard optimization techniques such as stochastic gradient descent or variants like Adam.

Practical Challenges and Enhancements

In practice, Deep Q-learning can be sensitive to the settings of its hyperparameters, including the learning rate, the size of the replay buffer, the batch size, and the frequency of target network updates. Enhancements like Double DQN, Dueling DQN, and Prioritized Experience Replay have been proposed to address these challenges and improve the robustness and performance of the base DQN algorithm.

These components together facilitate the Deep Q-learning algorithm's ability to efficiently learn policies in complex, high-dimensional environments, thereby enabling applications that range from playing video games at a superhuman level to controlling robots in dynamic real-world settings.

5.3.2 Deep Q-network (DQN) in pair trading

To incorporate Deep Q-Networks (DQNs) into pairs trading, one must develop a sophisticated approach that leverages deep reinforcement learning (RL) to optimize sequential decision-making processes. This paper outlines a comprehensive method for applying DQNs in pairs trading, including the setup of the trading environment, the design of the network architecture, and the fine-tuning of the learning algorithms.

Paper [53] outlines a structured approach for employing DQN networks to pinpoint and leverage opportunities in pairs trading. The following are detailed procedures drawn from the insights of the article.

Defining the Trading Strategy and Environment Setup

Pairs trading involves identifying two co-integrated stocks whose price spreads are expected to revert to a mean. The DQN's objective is to learn the temporal dynamics of this spread and predict optimal trading points.

Environment Construction:

- OpenAI Gym Framework: The trading environment is built using the OpenAI Gym framework, which offers a standardized API for reinforcement learning algorithms. This custom environment simulates the trading process, providing state observations that include historical price data and derived indicators.
- State Representation: The state provided to the DQN includes derived features such as moving averages, price ratios, historical volatility, and other statistical measures that represent the financial dynamics of the pair.

Data Handling and Feature Engineering

High-quality data preprocessing and feature engineering are critical for the performance of any ML model, particularly in complex domains like financial markets.

Preprocessing Steps:

- Dynamic Features: The features fed into the DQN include not only static indicators like price ratios but also dynamic measures that capture the recent behavior of the spread, such as short-term volatility and the derivative of the spread.
- Normalization and Standardization: All features are normalized or standardized to have a mean of zero and a standard deviation of one before being input into the network. This preprocessing step ensures that all features contribute equally to the model's decision process, preventing any one feature from dominating due to scale differences..

Defining Actions and Reward Mechanism

The DQN must decide among three actions based on the observed state, which aims to maximize the expected future rewards by exploiting the mean-reversion property of the spread.

Action Space:

- Buy and Short (Go Long): When the spread exceeds the upper threshold.
- Short and Buy (Go Short): When the spread falls below the lower threshold.
- Hold/No Action: When the spread does not meet either condition.

Reward Function:

- Immediate Reward: Calculated as a function of the profit or loss resulting from an action taken based on the spread movement. This might include a factor of immediate return multiplied by the spread's deviation from a moving average.
- Negative Returns Multiplier: During training, negative rewards are scaled up by a multiplier to penalize undesirable actions more severely. This approach teaches the DQN to avoid risky or unprofitable trades, promoting a more conservative strategy when uncertainty is high.

Network Architecture and Training Process

The DQN architecture typically involves several layers designed to process sequential data and predict the optimal action based on the current state.

Neural Network Design:

- Input Layer: Consists of 10 features representing the current state of the trading environment. These features include the spread between the two stocks, moving averages, volatility measures, and possibly other indicators like momentum or rate of change that provide insights into the price dynamics of the pairs.
- Hidden Layers: The network employs several hidden layers, typically two or three, each containing a substantial number of neurons (e.g., 50-100). These layers use Rectified

Linear Unit (ReLU) activation functions to introduce non-linearity into the model, allowing it to capture complex patterns in the data.

- Output Layer: The final layer of the network outputs the Q-values for each possible action: long, short, or no position. This direct mapping from state features to action values allows the DQN to evaluate the potential reward of each action based on the current market conditions.

Training Dynamics:

- Experience Replay: The model utilizes an experience replay buffer to store transitions, which are tuples of (current state, action taken, reward received, next state). This method helps to break the correlation between consecutive training samples, thereby stabilizing the learning process.
- Batch Sampling: Periodically, batches of transitions are sampled randomly from the replay buffer to train the network. This approach ensures that the network experiences a diverse range of situations, enhancing its ability to generalize across different market scenarios.

Q-value Calculation and Optimization:

- Loss Function: The network is trained using a mean squared error loss function, which measures the discrepancy between the predicted Q-values and the target Q-values calculated using the Bellman equation:

  $$Target\ Q - value = reward + \gamma \times max(next\ state\ Q - values)$$

  where γ is the discount factor, emphasizing the importance of future rewards.

- Optimization Algorithm: The Adam optimizer is employed for adjusting the network's weights. Adam is chosen for its efficiency in handling sparse gradients and its adaptive learning rate capabilities, which are ideal for complex environments like financial markets.

Evaluation and Continuous Learning

After training, the DQN is tested in a simulated environment using unseen data from the subsequent year (2018).

Testing:

- Performance Metrics: Includes total cumulative returns, the accuracy of spread predictions, and the number of successful trades versus unsuccessful ones.
- Consistency Check: Evaluates whether the strategy performs consistently across different market conditions and over time.

Optimization:

- Hyperparameter Tuning: Adjusting learning rate, size of the replay buffer, and the architecture of the neural network to optimize performance.

- Feedback Loop: Incorporating feedback from the testing phase to refine the model, such as adjusting the thresholds for actions or redefining the reward structure.

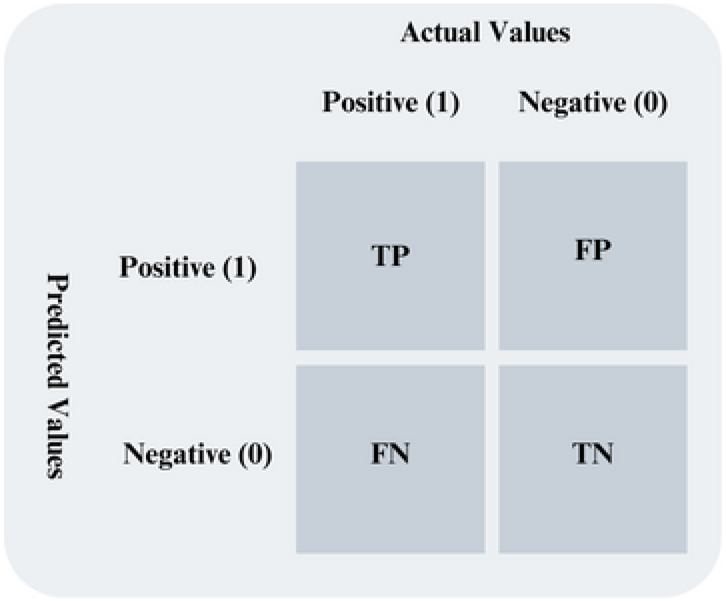
By carefully defining the trading strategy, preparing the data, setting up the neural network, and iterating through training and testing phases, this detailed implementation of DQN for pairs trading captures the complex dynamics of financial markets and seeks to harness them in a profitable trading strategy. This deep dive into the mechanics and methodology provides a robust framework for leveraging deep reinforcement learning in high-stakes financial applications.

Evaluation Metric

Two distinct categories of performance evaluation metrics are frequently utilized. The initial category appraises the effectiveness of the model (as seen in Table 18), while the second assesses the performance of the portfolio (noted in Table 19). To gauge the accuracy of classification models over a specified test dataset, a confusion matrix is employed. Figure 7 illustrates such a matrix for binary classification, which encompasses just two outcomes: positive and negative.

In pairs trading, utilizing a confusion matrix can offer valuable insights into the performance of a trading strategy, especially when the strategy involves classification-based decision-making, such as predicting whether the spread between two assets will converge or diverge. The confusion matrix is a tool that helps quantify the accuracy of a predictive model by comparing its predicted classifications against actual outcomes.

Figure 7. Confusion matrix



A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one, such as classification. It is especially useful for measuring recall, precision, specificity, accuracy, and other metrics in a binary or multiclass classification problem.

For a binary classification problem, the confusion matrix consists of two rows and two columns that report the number of:

True Positives (TP): These are cases in which the model correctly predicted the positive class.

True Negatives (TN): These are cases in which the model correctly predicted the negative class.

False Positives (FP): These are cases in which the model incorrectly predicted the positive class (also known as a "Type I error").

False Negatives (FN): These are cases in which the model failed to predict the positive class (also known as a "Type II error").

6.1 Evaluation Metrics of Efficacy

In assessing the efficacy of models produced by machine learning models, it is crucial to apply appropriate evaluation metrics that provide insights into their predictive accuracy and reliability. These metrics facilitate an understanding of how well the models predict new data, based on known outcomes. Let Y_i represent the ith actual value and $\hat{Y}_i$ denote the ith predicted value, where N is the total number of predictions made.

$$Y_i = ith\ actual\ value$$

$$\hat{Y}_i = ith\ predicted\ value$$

Table 18. Evaluation metrics for assessing model efficacy in reviewed articles

Evaluation Metrics	Description	Formula	Article
Mean Absolute Error (MAE)	Measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the mean over the test sample of the absolute differences between predicted values and actual values.	$MAE = \frac{1}{N}\sum_{i=1}^{N} \lvert Y_i - \hat{Y}_i \rvert$	[20, 45]
Mean Squared Error (MSE)	The average squared difference between the estimated values and the actual value. MSE is more sensitive to outliers than MAE as it squares the differences.	$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$	[16, 20, 25, 37, 45, 60]
Root Mean Squared Error (RMSE)	RMSE is the square root of the mean of the squared errors. RMSE is particularly useful when large errors are particularly undesirable.	$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}$	[23, 28, 40, 45,56, 65]

Mean Absolute Percentage Error (MAPE)	MAPE is a measure of prediction accuracy of a forecasting method in statistics. It usually expresses the accuracy as a ratio	$MAPE = \frac{100\%}{N}\sum_{i=1}^{N} \lvert\frac{Y_i - \hat{Y}_i}{Y_i}\rvert$	[16, 28, 31, 37, 40, 60]
Accuracy	Measures the overall effectiveness of the model in predicting correct outcomes.	$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$	[14, 24, 51, 52, 64]
Error rate	It quantifies the frequency at which the model accurately forecasts the outcome.	$Error\ rate = \frac{FP + FN}{TP + TN + FP + FN}$	[48]
Precision	Evaluates the reliability of the model in predicting positive (converging) outcomes.	$Precision = \frac{TP}{TP + FP}$	[14, 24, 52, 64]
Recall	Assesses the model's capability to identify actual positive (converging) instances.	$Recall = \frac{TP}{TP + FN}$	[14, 24, 52, 64]
F1-Score	Balances precision and recall, particularly useful when the cost of false positives and false negatives are high.	$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$	[14, 24, 52, 64]
AUC	AUC represents the area under the ROC curve and ranges from 0 to 1. The higher the AUC value, the better the classification performance of the model.	N.A.	[30, 64]

6.2 Performance of Evaluation Metrics

Evaluating the performance of an investment portfolio is a critical step in the investment decision-making process. The primary dimensions of this evaluation are return and risk, which serve as the foundational metrics for assessing portfolio outcomes. A variety of specific indicators, falling under these overarching categories, are employed to monitor and analyze the progression of an investment portfolio's performance. **Table 19** provides examples of such performance and risk indicators that investors utilize for portfolio assessment.

Table 19. Evaluation metrics for assessing model performance in reviewed articles

Evaluation Metrics	Description	Article
Accumulated Return	Accumulated Return denotes the overall percentage gain or decline in the value of an investment throughout its duration. Ideally, the Accumulated Return should be positive and maximized.	[3, 6, 7, 8, 10, 17, 18, 29, 39, 42, 44, 53, 62, 65]
Annual Return	Annual return is the percentage change in the value of an investment over a one-year period, reflecting the compound rate of	[2, 4, 5, 8, 12, 22, 25, 31, 32, 33, 35, 39, 41, 46, 47, 55, 58, 59, 63, 68]

	return earned or lost by the investment annually.	
Sharpe ratio (SR)	The Sharpe ratio is a measure used to evaluate the risk-adjusted return of an investment by comparing its excess return over the risk-free rate to the standard deviation of those returns.	[2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 14, 15, 18, 20, 21, 22, 26, 29, 31, 33, 34, 38, 39, 40, 41, 43, 44, 45, 47, 49, 50, 55, 58, 62, 65, 66, 68]
Maximum Drawdown (MDD)	Maximum Drawdown is a risk measure that indicates the largest single drop from peak to trough in the value of a portfolio or an investment, before a new peak is achieved.	[2, 3, 5, 6, 8, 9, 10, 12, 15, 18, 20, 21, 22, 25, 26, 27, 29, 31, 33, 35, 38, 39, 41, 43, 45, 47, 58, 61, 64, 68]
Standard Deviation (Volatility)	quantifies the variability or dispersion of a set of data points from their mean, typically used to assess the consistency of an investment's returns.	[2, 4, 5, 8, 12, 33, 36, 38, 40, 41, 55, 58, 61, 62, 63, 65, 66, 68]
Skewness	Refers to the asymmetry in the distribution of returns for an investment, indicating whether the returns are biased towards higher or lower values than the average, which can reveal the potential for unpredictable extreme outcomes in investment performance.	[4, 58, 61, 62]
Kurtosis	Measures the "tailedness" of the return distribution of an investment, indicating the likelihood of extreme positive or negative returns compared to a normal distribution.	[4, 58, 61, 62]
Sortino ratio	Measures the excess return of an investment relative to the downward deviation, focusing specifically on the volatility of negative asset returns, thus providing a risk-adjusted measure of a portfolio's performance under negative fluctuations.	[9, 10, 20, 21, 26, 29, 41, 55, 58, 63]

7. Data Availability and Implementation

For conducting research aimed at forecasting stock market trends, the internet is a treasure trove of freely available data sources. Yahoo! Finance, widely utilized for its no-cost access to stock quotes, current market news, and international market statistics, is highly recommended and has been referenced in 25 out of 68 studies. It is particularly useful for those seeking historical price and volume data. Additionally, platforms like Kaggle, which host data science competitions, are becoming increasingly popular as resources for stock market prediction datasets. Quantitative firms often collaborate with these platforms to sponsor competitions focused on developing stock market forecasts. Some useful data source links include: https://finance.yahoo.com, https://github.com/, https://www.kaggle.com/, https://www.nseindia.com/, and https://www.wikipedia.org/ (accessed on 15 April 2024).

8. Empirical Result Analysis and Future Research Direction

8.1 Empirical Result Analysis of Pair Trading

At present, pair trading using algorithmic trading becomes a mainstream trading technique in quantitative finance domain. More and more investors including institutions and individuals pour in the market and star arbitraging, which makes the financial market increasingly competitive and hard to explore arbitrage opportunities. Therefore, finding an effective approach to explore arbitrage opportunities is quite necessary to ensure sufficient profits. [14]

Upon reviewing and synthesizing the literatures, it has been determined that pairs trading remains a viable and profitable strategy, despite a minority of studies suggesting otherwise.

The neural network's ability to uncover non-linear relationships and identify intricate patterns enhances the precision and effectiveness of trading signal predictions, providing traders with a valuable tool for informed decision-making. [1]

LSTM can achieve good performance in overall accuracy and the different metrics in positive arbitrage samples. CNN is more sensitive the non-arbitrage samples and it achieve higher precision and f-measure in non-arbitrage samples. Logistic regression performs relatively balanced in different samples. Concerning the profits analysis, machine learning strategy outperform than cointegration strategy. Specifically, LSTM obtain highest annualized ROI and explore most arbitrage opportunities than other strategies. XGBoost has the maximum positive average ROI even though it explores the least arbitrage opportunities, which means that XGBoost performs better in arbitrage timing. The profitability of machine learning strategy further substantiates the market efficiency. [14]

By integrating the Entropy Optimization Criterion (EOC) for the selection of trading pairs and the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) method for establishing trading thresholds. The efficacy of this comprehensive approach is demonstrated through rigorous simulations conducted within the Chinese futures market. Our results indicate significant improvements: employing the DQN for pair selection at fixed intervals led to a 50% increase in Cumulative Return (CR) compared to the conventional Static-Average method. Further enhancements were achieved by implementing the EOC method, which dynamically determines the optimal times for trade closure, resulting in a 23% CR increase over the DQN method. Additionally, the adoption of the MADDPG method, which utilizes cooperative communication among agents, provided a CR improvement ranging from 5% to 15% over DQN. The integration of both EOC and MADDPG methods yielded a remarkable 29% improvement in CR over static methods and a 16% improvement over the DQN method alone, underscoring the potential of this dual approach to significantly optimize pairs trading strategies. [3]

TCN has proven more effective in stock price prediction than traditional financial algorithms, offering substantial profit gains in pairs trading. Extensions of TCN, such as the Knowledge-Driven Stock Trend Prediction (KDTCN) and Temporal Convolutional Attention-based Network

(TCAN) that integrate natural language processing (NLP), promise even greater accuracy improvements. [28]

However, a few papers find that pairs trading is no longer profitable.

Even if we consider fundamentals similarity, the performance of pairs trading has significantly declined since 2003. This indicates that arbitrage opportunities have disappeared as the market became more efficient. However, the cost of trading spurious pairs is still significant despite the recent poor performance of pairs trading. This means that fundamental is still an important factor to identify reliable pairs and arbitrageurs should identify spurious pairs using fundamentals. [4]

8.2 Future Research Direction

In the rapidly evolving field of algorithmic pairs trading, where machine learning and deep reinforcement learning are integral, the imperative to expand and refine methodologies is clear. This pursuit involves not only enhancing existing strategies but also exploring innovative applications and addressing the intricate challenges associated with sophisticated financial models. Here, we delineate a comprehensive roadmap for future research, aiming to push the boundaries of technology and methodology in financial trading.

Comprehensive Multi-Modal Data Integration

Future research should aim to transcend traditional market data by incorporating a diverse array of unstructured data sources such as geopolitical news, economic reports, and real-time social media feeds. The goal is to develop multi-modal models that leverage advanced text analysis, sentiment analysis, and image recognition techniques. These models would interpret sentiment and thematic trends from vast datasets to improve the predictive accuracy of trading algorithms significantly.

Sector-Specific Model Development

There is considerable potential in tailoring models to specific industries, given their unique market dynamics and risk profiles. Future studies should focus on customizing feature engineering and model architectures to accommodate the behavioral finance aspects of sectors such as technology, energy, or pharmaceuticals. This approach could yield more precise predictions and, by extension, more robust trading strategies.

Advanced Anomaly Detection Techniques

Implementing cutting-edge statistical and machine learning techniques for anomaly detection in financial time series is crucial. These methods would identify potential market manipulations or flash crashes in real-time, thus safeguarding trading strategies against significant unpredictable risks. Developing dynamic monitoring systems that react instantaneously to unusual market movements will be a key area of focus.

Algorithmic Adaptation to Market Conditions

Enhancing models to adapt dynamically to changing market conditions without manual intervention is another critical research direction. This includes developing adaptive algorithms that can modify their decision-making processes based on real-time market sentiment or volatility indices. Such systems would offer significant advantages in rapidly changing market environments.

Game Theory and Agent-Based Models

The application of game theory and agent-based models to simulate interactions among multiple trading agents could provide deep insights into market dynamics and trader behaviors. This research would involve using game theory to predict the actions of market participants and developing simulations that study the collective outcomes of various trading strategies.

Ethical AI and Compliance Automation

As machine learning becomes more prevalent in trading, ensuring these systems adhere to ethical standards and regulatory requirements is paramount. Future research should focus on developing compliance automation tools that monitor and ensure trading activities remain within legal and ethical boundaries.

Deep Reinforcement Learning Enhancements

Integrating state-of-the-art techniques such as meta-learning within deep reinforcement learning frameworks could allow agents to learn more effectively across a wide range of tasks by adapting quickly to new environments. Such enhancements would be particularly useful in financial markets where conditions change rapidly.

Cross-Asset Arbitrage and Intermarket Strategies

Developing and testing cross-asset arbitrage strategies that involve pairs trading not only within but also across asset classes, such as equities, commodities, and currencies, is an ambitious yet potentially rewarding endeavor. This requires building complex models that understand and can capitalize on the relationships between different types of assets.

Quantum Computing for Optimization

Exploring the potential of quantum computing to optimize trading algorithms could revolutionize high-frequency trading strategies. Quantum algorithms may solve complex optimization problems faster than classical algorithms, offering significant advantages in terms of speed and efficiency.

Real-World Implementation and Back-testing

Conducting extensive back-testing using historical data as well as forward-testing in simulated environments is essential to validate the effectiveness of newly developed trading models. This should include stress testing under various hypothetical scenarios to evaluate the robustness of the strategies.

By addressing these detailed areas, researchers and practitioners can develop more sophisticated, robust, and adaptive trading strategies that not only enhance profitability but also contribute to market stability and investor confidence. This comprehensive approach to research and development in algorithmic pairs trading aims to ensure that the financial trading landscape remains both innovative and secure.

9. Discussion

In our comprehensive investigation, we meticulously dissected the evolution of pair trading strategies, with a keen focus on the burgeoning integration of artificial intelligence (AI) methodologies. Our journey commenced with an exhaustive review of conventional techniques deeply rooted in time-series forecasting principles. These foundational methods, characterized by their reliance on statistical analysis and linear models, provided the bedrock upon which subsequent innovations were built.

As the field of machine learning rapidly advanced, we bore witness to a seismic shift towards more sophisticated algorithms capable of unraveling the intricacies embedded within financial data. Among the pantheon of these groundbreaking approaches, the random forest classifier and the support vector machine (SVM) emerged as stalwarts, showcasing remarkable prowess in deciphering complex nonlinear relationships inherent in financial time series. Notably, the support vector machine algorithm's ability to delineate optimal hyperplanes through intricate data transformations into higher-dimensional spaces underscored its significance in discerning nuanced market dynamics.

Nevertheless, the advent of artificial neural networks (ANNs) marked a watershed moment in financial modeling. Inspired by the human brain's neural architecture, ANNs revolutionized predictive analytics by adeptly capturing latent patterns and trends embedded within vast troves of financial data. Leveraging their innate ability to generalize from historical data, ANNs swiftly cemented their status as indispensable tools in quantitative finance applications. Yet, recent years have witnessed an unprecedented surge in the adoption of recurrent neural networks (RNNs) and their long short-term memory (LSTM) variants, owing to their unparalleled capacity to capture temporal dependencies intrinsic to sequential data.

Venturing beyond the realm of static forecasting models, our study ventured into the uncharted territory of dynamic trading strategies facilitated by reinforcement learning (RL) paradigms. Harnessing the principles of iterative trial-and-error learning, RL algorithms empower traders to iteratively refine their decision-making processes based on real-time feedback from the ever-evolving market landscape. This iterative adaptation enables traders to dynamically adjust their strategies in response to shifting market conditions, thereby optimizing performance and mitigating risk.

Furthermore, recent strides in research have seen a convergence of deep learning (DL) and RL techniques, culminating in hybrid approaches that amalgamate the feature extraction capabilities of deep neural networks with the adaptive learning prowess of RL algorithms. This synergistic fusion has yielded unprecedented insights into the complex dynamics of financial markets, enabling traders to devise robust trading strategies capable of navigating turbulent market environments with unparalleled dexterity.

In summation, our study underscores the transformative potential of AI in reshaping pair trading strategies. By harnessing the predictive power of machine learning models and the adaptive learning capabilities of RL algorithms, traders stand poised to gain a decisive edge in today's hypercompetitive financial landscape. However, continued research endeavors are imperative to unravel the intricacies of implementing these advanced techniques in real-world trading scenarios and to mitigate potential risks inherent in algorithmic trading.

10. Conclusion

The final section of this survey comprehensively presents the practical and theoretical implications of our findings. It introduces a systematic process designed for those new to the area, providing a step-by-step guide to ensure ease of understanding and application. The incorporation of previously reviewed articles as standard benchmarks invites crucial discussions about their application strategies and the possibility of replicating their results, which is pivotal for advancing the field.

From a theoretical perspective, our research zeroes in on machine learning and deep learning. These technologies were chosen due to their broad applicability and demonstrated success in significantly enhancing the capabilities of expert and intelligent systems. This paper investigates the cutting-edge developments in deep learning as applied to specific scenarios, such as forecasting movements in the stock market, and offers a synthesis of the latest findings.

In addition, a primer on the fundamental theories underlying machine learning and deep learning is provided, furnishing readers with the necessary conceptual tools to engage with these technologies. The paper also identifies and outlines promising avenues for further research, aiming to inspire and guide future scholars interested in this dynamic field.

When applying machine learning, deep learning, and reinforcement learning to stock market prediction, the efficacy of these models depends crucially on the quality and relevance of the input data reflective of current market conditions. It is important to recognize that the same models may not yield identical results or maintain the same efficiency with future datasets. This variability underscores the inherently unpredictable nature of stock markets, which are influenced by a complex interplay of social, political, economic, and demographic factors. Each of these influences can sway market behavior, making the task of prediction particularly challenging yet fascinating.

Abbreviations

ADR - Average Daily Return

AHC - Hierarchical Clustering Algorithm

AI - Artificial Intelligence

AMEX - American Stock Exchange

ANFIS - Adaptive Neuro Fuzzy Inference System

ANN - Artificial Neural Network

ATS - Automated Trading System

AUC - Area Under the Curve

AdaBoost - Adaptive Boosting

BNN - Bayesian Neural Network

BNN - Bayesian Neural Network

BP - Back Propagation

BPTT - Backpropagation through time

CA-DRL - Cointegration Approach-Deep Reinforcement Learning

CAE - Convolutional AutoEncoders

CNN - Convolutional Neural Network

DBSCAN - Density-Based Spatial Clustering of Applications with Noise

DCE - Dalian Commodity Exchange

DDQN - Double Deep Q-Network

DL - Deep Learning

DPG - Deep Policy Gradient

DPG - Deterministic Policy Gradient

DQN - Deep Q-network

DRL - Deep Reinforcement Learning

DT - Decision Tree

ELM - Extreme Learning Machine

EM - Expectation Maximization

EM - Expectation Maximization

EMH - Efficient Market Hypothesis

EN - Elastic Net

EOC - Extended Option-Critic

FN - False Negatives

FP - False Positives

GBDT - Gradient Boosted Decision Trees

GBT - Gradient-boosted Tree

GDA - Gaussian Discriminant Analysis

HDRL - Hierarchical Deep Reinforcement Learning

HFT - High-Frequency Trading

KalmanBOT - KalmanNet Bollinger Trading

LGBM - LightGBM

LR - Logistic Regression

LSTM - Long Short-Term Memory

MADDPG - Multi-Agent Deep Deterministic Policy Gradient

MAE - Mean Absolute Error

MAPE - Mean Absolute Percentage Error

MDD - Maximum Drawdown

MDP - Markov Decision Process

ML - Machine Learning

MLP - Multilayer perceptron

MSE - Mean Squared Error

NB - Naive Bayes

NLP - Natural Language Processing

NN - Neural Network

NYSE - New York Stock Exchange

O-U - Ornstein-Uhlenbeck

OHLCV - Open, High, Low, Close and Volume

PCA - Principal Component Analysis

PPO - Proximal Policy Optimization

PSX - Pakistan Stock Exchange

QT - Quantitative Trading

RBF - Radial Basis Function

RF - Random Forest

RL - Reinforcement Learning

RMSE - Root Mean Squared Error

RNN - Recurrent Neural Network

ROC - Receiver Operating Characteristic

ReLU - Rectified Linear Unit

SAPT - Structural break-aware pairs trading strategy

SAPT - Structured Approach to Pair Trading

SNN - Stochastic Neural Network

SR - Sharpe ratio

SSE - Shanghai Stock Exchange

SVM - Support Vector Machine

SVR - Support Vector Regression

TAIEX - Taiwan Stock Exchange Capitalization Weighted Stock Index

TCN - Temporal Convolutional Network

TN - True Negatives

TP - True Positives

XGBoost - eXtreme Gradient Boosting

References[2]

[1] Platania, F., Appio, F., Toscano Hernandez, C., El Ouadghiri, I. and Peillex, J., 2023. A multi-objective pair trading strategy: integrating neural networks and cyclical insights for optimal trading performance. Annals of Operations Research, pp.1-20.

[2] He, J., Liu, J., Fu, Y., Zhu, Y., Zhou, Z. and Tong, Y., 2023. Applications of Copulas and Machine Learning Algorithms on Pairs Trading.

[3] Xu, Z. and Luo, C., 2023. Improved pairs trading strategy using two-level reinforcement learning framework. Engineering Applications of Artificial Intelligence, 126, p.107148.

[4] Hong, S. and Hwang, S., 2023. In search of pairs using firm fundamentals: is pairs trading profitable?. The European Journal of Finance, 29(5), pp.508-526.

[5] Liu, J., Lu, L., Zong, X. and Xie, B., 2023. Nonlinear relationships in soybean commodities Pairs trading-test by deep reinforcement learning. Finance Research Letters, 58, p.104477.

[6] Liu, J., Lu, L., Zong, X. and Luo, Z., 2023. Nonlinear Relationships in Stock News Co-occurrence: A Pairs Trading Test on the Constituent Stocks of the CSI 300 Index Based on Deep Reinforcement Learning Methods. Available at SSRN 4573596.

[7] Gupta, V., Kumar, V., Yuvraj, Y. and Kumar, M., 2023, April. Optimized pair trading strategy using unsupervised machine learning. In 2023 IEEE 8th International Conference for Convergence in Technology (I2CT) (pp. 1-5). IEEE.

[8] Roychoudhury, R., Bhagtani, R. and Daftari, A., 2023. Pairs Trading Using Clustering and Deep Reinforcement Learning. Available at SSRN 4504599.

[9] Han, C., He, Z. and Toh, A.J.W., 2023. Pairs trading via unsupervised learning. European Journal of Operational Research, 307(2), pp.929-947.

[10] Chen, Y.F., Shih, W.Y., Lai, H.C., Chang, H.C. and Huang, J.L., 2023, February. Pairs Trading Strategy Optimization Using Proximal Policy Optimization Algorithms. In 2023 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 40-47). IEEE.

[11] Afzal, M., Usman, M. and Raheman, A., 2023. Comparative Study of Pair Trading Techniques in Pakistan's Financial and Non-Financial Sector. Reviews of Management Sciences, 5(1), pp.38-49.

[12] Han, W., Zhang, B., Xie, Q., Peng, M., Lai, Y. and Huang, J., 2023, August. Select and trade: Towards unified pair trading with hierarchical reinforcement learning. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 4123-4134).

[13] Sohail, M.K., Raheman, A., Iqbal, J., Sindhu, M.I., Staar, A., Mushafiq, M. and Afzal, H., 2022. Are Pair Trading Strategies Profitable During COVID-19 Period?. Journal of Information & Knowledge Management, 21(Supp01), p.2240010.

[2] Since the order in which the papers are used in the article is chronological, they are not sorted alphabetically.

[14] Zhan, B., Zhang, S., Du, H.S. and Yang, X., 2022. Exploring statistical arbitrage opportunities using machine learning strategy. Computational Economics, 60(3), pp.861-882.

[15] Kim, S.H., Park, D.Y. and Lee, K.H., 2022. Hybrid deep reinforcement learning for pairs trading. Applied Sciences, 12(3), p.944.

[16] Zhijie, Z., 2022. Identify Arbitrage Using Machine Learning on Multi-stock Pair Trading Price Forecasting (Doctoral dissertation, Tohoku University).

[17] Deng, H., Revach, G., Morgenstern, H. and Shlezinger, N., 2023, June. Kalmanbot: Kalmannet-Aided Bollinger Bands for Pairs Trading. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.

[18] Sviridenko, E.V., Filipchenkov, V.D. and Zuyonok, R.V., 2022. Machine learning algorithms in pair trading.

[19] Figueira, M. and Horta, N., 2022. Machine Learning-Based Pairs Trading Strategy with Multivariate. Available at SSRN 4295303.

[20] Du, J., 2022. Mean–variance portfolio optimization with deep learning based-forecasts for cointegrated stocks. Expert Systems with Applications, 201, p.117005.

[21] Zhang, Y., 2022. Multivariate Bilateral Gamma Process with Financial Application and Machine Learning in Corporate Bond Market (Doctoral dissertation, University of Maryland, College Park).

[22] Huang, K., Sun, J., Zhang, Z. and Li, Q., 2022. Pair trading of Commodity Futures in China through the lens of intraday data A Machine Learning Framework and Empirical Analysis. Available at SSRN 4117105.

[23] Zhang, Y., Shen, Q., Guo, J. and Jia, Y., 2022, September. Portfolio Trading of Financial Products Based on Machine Learning. In 2022 International Conference on Machine Learning and Cybernetics (ICMLC) (pp. 72-79). IEEE.

[24] Su, C.H., Lai, H.C., Shih, W.Y., Wang, J.Z. and Huang, J.L., 2022, January. Spread Movement Prediction for Pairs Trading with High-Frequency Limit Order Data. In 2022 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 64-71). IEEE.

[25] Carta, S., Consoli, S., Podda, A.S., Recupero, D.R. and Stanciu, M.M., 2022. Statistical arbitrage powered by explainable artificial intelligence. Expert Systems with Applications, 206, p.117763.

[26] Zhang, M., Tang, X., Zhao, S., Wang, W. and Zhao, Y., 2022. Statistical arbitrage with momentum using machine learning. Procedia Computer Science, 202, pp.194-202.

[27] Kalariya, V., Parmar, P., Jay, P., Tanwar, S., Raboaca, M.S., Alqahtani, F., Tolba, A. and Neagu, B.C., 2022. Stochastic neural networks-based algorithmic trading for the cryptocurrency market. Mathematics, 10(9), p.1456.

[28] Li, J., Wang, S., Zhu, Z., Liu, M., Zhang, C. and Han, B., 2022, July. Stock Prediction Based on Deep Learning and its Application in Pairs Trading. In 2022 International Symposium on Networks, Computers and Communications (ISNCC) (pp. 1-7). IEEE.

[29] Lu, J.Y., Lai, H.C., Shih, W.Y., Chen, Y.F., Huang, S.H., Chang, H.H., Wang, J.Z., Huang, J.L. and Dai, T.S., 2022. Structural break-aware pairs trading strategy using deep reinforcement learning. The Journal of Supercomputing, 78(3), pp.3843-3882.

[30] Yu, Z., 2022. The Implementation of Support Vector Machine into Pairs Trading Strategy. Frontiers in Business, Economics and Management, 4(3), pp.159-165.

[31] Relan, T., 2021. Applying Machine Learning Techniques to Assess Whether a Country's Currency can Predict the Movement of their Respective Stock Market Index.

[32] Patole, D., Gupta, I., Jain, P., Gupta, V. and Gada, Y., 2021, September. Controlled Risk Pairs Trading using ReinforcementLearning. In 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 1-5). IEEE.

[33] Zong, X., Liu, J. and Wang, C., 2021. Deep Reinforcement Learning for Pairs Trading: Evidence from Soybean Commodities. Available at SSRN 3831581.

[34] Wang, C., Sandås, P. and Beling, P., 2021, May. Improving pairs trading strategies via reinforcement learning. In 2021 International Conference on Applied Artificial Intelligence (ICAPAI) (pp. 1-7). IEEE.

[35] Zong, X., 2021. Machine learning in stock indices trading and pairs trading (Doctoral dissertation, University of Glasgow).

[36] Zhang, L., 2021, May. Pair trading with machine learning strategy in china stock market. In 2021 2nd International Conference on Artificial Intelligence and Information Systems (pp. 1-6).

[37] Chang, V., Man, X., Xu, Q. and Hsu, C.H., 2021. Pairs trading on different portfolios based on machine learning. Expert Systems, 38(3), p.e12649.

[38] Flori, A. and Regoli, D., 2021. Revealing pairs-trading opportunities with long short-term memory networks. European Journal of Operational Research, 295(2), pp.772-791.

[39] Carta, S., Recupero, D.R., Saia, R. and Stanciu, M.M., 2020, July. A general approach for risk controlled trading based on machine learning and statistical arbitrage. In International Conference on Machine Learning, Optimization, and Data Science (pp. 489-503). Cham: Springer International Publishing.

[40] Osifo, E. and Bhattacharyya, R., 2020. Cryptocurrency trading-pair forecasting, using machine learning and deep learning technique. Using Machine Learning and Deep Learning Technique (March 5, 2020).

[41] Sermpinis, G., Stasinakis, C. and Zong, X., 2020. Deep reinforcement learning and genetic algorithm for a pairs trading task on commodities. Available at SSRN 3770061.

[42] Brim, A., 2020, January. Deep reinforcement learning pairs trading with a double deep Q-network. In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0222-0227). IEEE.

[43] Navghare, N.D., Kulkarni, H.P., Kedar, P.R., Thakre, S.A. and Patil, P.S., 2020, July. Design of pipeline framework for pair trading algorithm. In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 630-635). IEEE.

[44] Xu, F. and Tan, S., 2020, November. Dynamic portfolio management based on pair trading and deep reinforcement learning. In Proceedings of the 2020 3rd International Conference on Computational Intelligence and Intelligent Systems (pp. 50-55).

[45] Sarmento, S.M. and Horta, N., 2020. Enhancing a pairs trading strategy with the application of machine learning. Expert Systems with Applications, 158, p.113490.

[46] Ospino, J.F., Fernandez, L. and Carchano, O., 2020. Improving pairs trading using neural network techniques and fundamental ratios. Available at SSRN 3653071.

[47] Baek, S., Glambosky, M., Oh, S.H. and Lee, J., 2020. Machine learning and algorithmic pairs trading in futures markets. Sustainability, 12(17), p.6791.

[48] Huang, S.H., Shih, W.Y., Lu, J.Y., Chang, H.H., Chu, C.H., Wang, J.Z., Huang, J.L. and Dai, T.S., 2020, February. Online structural break detection for pairs trading using wavelet transform and hybrid deep learning model. In 2020 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 209-216). IEEE.

[49] Sohail, M.K., Raheman, A., Adil, I.H., Rizwan, M.F. and Khan, S.U., 2020. Pair Trading Strategies Using Machine Learning: A Case of PSX Firms. Pakistan Business Review, 22(3), pp.340-351.

[50] Yoshikawa, D., 2020. Pairs trading with deep learning. In the 34th Annual Conference of the Japanese Society for Artificial Intelligence (2020) (pp. 3I1GS1301-3I1GS1301).

[51] Jirapongpan, R. and Phumchusri, N., 2020, April. Prediction of the profitability of pairs trading strategy using machine learning. In 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA) (pp. 1025-1030). IEEE.

[52] Fuster, A. and Zou, Z., 2020. Using Machine Learning Models to Predict S&P500 Price Level and Spread Direction.

[53] Brim, A., 2019. Deep reinforcement learning pairs trading.

[54] Sun, J., Zhou, Y. and Lin, J., 2019, May. Using machine learning for cryptocurrency trading. In 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS) (pp. 647-652). IEEE.

[55] Fischer, T.G., Krauss, C. and Deinert, A., 2019. Statistical arbitrage in cryptocurrency markets. Journal of Risk and Financial Management, 12(1), p.31.

[56] Ruxanda, G. and Opincariu, S., 2018. BAYESIAN NEURAL NETWORKS WITH DEPENDENT DIRICHLET PROCESS PRIORS. APPLICATION TO PAIRS TRADING. Economic Computation & Economic Cybernetics Studies & Research, 52(4).

[57] Chen, Y.Y., Chen, W.L. and Huang, S.H., 2018, July. Developing arbitrage strategy in high-frequency pairs trading with filterbank CNN algorithm. In 2018 IEEE International Conference on Agents (ICA) (pp. 113-116). IEEE.

[58] Sutherland, I., Jung, Y. and Lee, G., 2018. Statistical arbitrage on the KOSPI 200: An exploratory analysis of classification and prediction machine learning algorithms for day trading. Journal of Economics and International Business Management, 6(1), pp.10-19.

[59] Li, Y., Bu, H. and Wu, J., 2017, June. A two-stage multi-view prediction method for investment strategy. In 2017 International Conference on Service Systems and Service Management (pp. 1-6). IEEE.

[60] Chaudhuri, T.D., Ghosh, I. and Singh, P., 2017. Application of Machine Learning Tools in Predictive Modeling of Pairs Trade in Indian Stock Market. IUP Journal of Applied Finance, 23(1).

[61] Krauss, C., Do, X.A. and Huck, N., 2017. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), pp.689-702.

[62] Kim, S. and Heo, J., 2017. Time series regression-based pairs trading in the Korean equities market. Journal of Experimental & Theoretical Artificial Intelligence, 29(4), pp.755-768.

[63] Fallahpour, S., Hakimian, H., Taheri, K. and Ramezanifar, E., 2016. Pairs trading strategy optimization using the reinforcement learning method: a cointegration approach. Soft Computing, 20, pp.5051-5066.

[64] Wu, J., 2015. A pairs trading strategy for GOOG/GOOGL using machine learning.

[65] Nóbrega, J.P. and Oliveira, A.L., 2014, October. A combination forecasting model using machine learning and kalman filter for statistical arbitrage. In 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 1294-1299). IEEE.

[66] Nóbrega, J.P. and De Oliveira, A.L.I., 2013, November. Improving the statistical arbitrage strategy in intraday trading by combining extreme learning machine and support vector regression with linear regression models. In 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (pp. 182-188). IEEE.

[67] Madhavaram, G.R., 2013. Statistical arbitrage using pairs trading with support vector machine learning.

[68] Chen, Y., Ren, W. and Lu, X., 2012. Machine Learning in Pairs Trading Strategies.