Correlated Stock Identification in Pairs Trading Using Extreme Gradient Boosting Algorithm

By:

Chrispus N. Muhia

53311

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Award of Degree in Master of Data Science and Analytics (MSc. DSA)

Strathmore Institute of Mathematical Sciences (SIMS)

Strathmore University

March 2024

Declaration

I confirm that I am the original author of this research proposal and that all non-original content has been properly cited and given credit to the original creator. I further declare that no other university has received the work submitted for a degree or other award. No part of this proposal may be reproduced without the express permission of the author and/or the Strathmore Institute of Mathematical Sciences (SIMS).

Signature...Chrispus N. Muhia..... Date …31 - 03 -2024……..…..…..

This dissertation has been submitted for examination with my approval as University Supervisor

Signature……*drBWMuganda*…… Date……2nd April 2024……..

Acknowledgement

I extend my heartfelt gratitude to Professor Yoon, Dr. Kennedy Senagi, and Dr. Brian Muganda for their invaluable support and guidance throughout the research process. Their expertise, encouragement, and insightful feedback have been instrumental in shaping the direction of this project. I also wish to acknowledge my classmates for their collaboration and knowledge sharing, which enriched my understanding of the subject matter. I especially want to thank Emmanuel Lee Bundi for his guidance in model evaluation and deployment. Additionally, I am deeply thankful to my family for their unwavering support and encouragement throughout this journey. Together, the support of my professors, the collaboration with my classmates, and the encouragement from my family have been indispensable in the successful completion of this research endeavor.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CCW: Cointegration Coefficients Weighted

NYSE: New York Stock Exchange

NASDAQ: National Association of Securities Dealers Automated Quotations

MSE: Mean Squared Error

MAE: Mean Absolute Error

MTP: minimum total profit

OLS: Ordinary Least Squares

PTS: Pairs Trading Strategy

SVM: Support Vector Machine

SOMs: Self-Organizing Maps

S&P 500: Standard & Poor's 500

XGBoost: eXtreme Gradient Boosting

Abstract

Pairs trading is a well-known market-neutral trading strategy that aims to exploit market inefficiencies by identifying and trading pairs of highly correlated stocks. This research addresses the pressing problem of accurately identifying correlated stock pairs for pairs trading strategies, recognizing the potential for reducing risk and generating profits in financial markets. While traditional statistical and deep learning methods have provided valuable insights, there exists a notable research gap in assessing the effectiveness of advanced machine learning algorithms like Extreme Gradient Boosting (XGBoost) in this context. To bridge this gap, the study meticulously compares the performance of the XGBoost algorithm with conventional techniques through quantitative analysis. Leveraging historical stock price data and machine learning methodologies, the research explores the intricacies of stock pairing accuracy and profitability. The findings reveal that the tuned XGBoost model demonstrates superior accuracy, precision, and recall in identifying profitable stock pairs, outperforming traditional statistical methods and other machine learning algorithms. Specifically, the XGBoost model achieved an accuracy of 95.50% and a precision of 95.34% in identifying profitable stock pairs. These results underscore the potential of XGBoost to enhance pairs trading strategies and optimize trading decisions in dynamic financial environments. However, while the XGBoost model showcases remarkable performance, it is not without limitations. Susceptibility to overfitting and reliance on input feature quality and quantity present challenges that need to be addressed. Nonetheless, the study provides valuable insights for investors and traders, suggesting avenues for optimizing trading strategies and maximizing profitability. Recommendations include further exploration of XGBoost's capabilities in diverse market conditions and the integration of additional data sources to enhance predictive accuracy. Moreover, the research highlights the need for continued investigation into other advanced machine learning algorithms and ensemble techniques to further improve stock pairing accuracy. Ultimately, this study contributes to advancing pairs trading strategies by providing empirical evidence of XGBoost's effectiveness, while also identifying avenues for future research and development in the field.

Key Words: Pairs Trading, Correlated Stocks, Autoencoders, Self-Organizing Maps, Random Forest, Support Vector Machine, Trading Strategy, Sharpe Ratio, Maximum Drawdown, Cointegration, Backtesting, Machine Learning, XGBoost

Chapter 1: Introduction

1.1 Background of the study

Stocks are monetary instruments that signify ownership in a company. They are frequently bought and sold on stock exchanges like the NASDAQ or the New York Stock Exchange (NYSE). Stock trading is the process of purchasing and disposing of stocks on the stock market with the goal of profiting. Analyzing financial statements from businesses, market swings, and other economic factors is necessary to determine when it is best to buy or sell stocks (Kapoor, Dlabay, and Hughes, 2017). Investing in stocks has the potential to yield substantial returns for investors. It also allows businesses to choose to raise capital through the sale of stock (Fama French, 2004). However, some of the major challenges associated with stock trading include the high degree of market volatility and uncertainty (Fama French, 2004).

The stock market is impacted by a multitude of factors, including shifts in geopolitics, the release of economic data, and unexpected corporate news (Kolb, 2010). Furthermore, past events such as the Wall Street Crash of 1929 and the Global Financial Crisis of 2008 have demonstrated the potential for significant losses in the stock market (Hilton, 2009). Despite these obstacles, stock trading is still a popular investment choice for individuals and businesses trying to increase capital for operations and earn returns on their capital. According to Fama and French (2004), the volatility and unpredictability of the stock market make stock trading a challenging and dynamic activity that can yield substantial gains or losses. One arbitrage strategy that can be applied to stock market trading is pairs trading. Finding stock pairs with strong correlations and profiting from their price differences is the main objective of pairs trading (Gatev, Goetzmann, and Rouwenhorst, 2006). For pairs trading strategies to be successful, it is essential to accurately identify correlated stock pairs (Guo and Zhu, 2016).

1.2 Pairs Trading

Pairs trading is a popular trading strategy in which two stocks with a high correlation are chosen, and long and short trading positions are taken on the stock pairs (Gatev et al., 2006). A pairs trading strategy (PTS), according to Kuo et al. (2022), creates and maintains a stationary portfolio by shorting (longing) a stock or portfolio when it hits a predetermined open threshold, indicating that the portfolio or stock is appropriately over- (under-)priced. Once the value of the portfolio reaches the mean level, we close this position to realize the price differences. The goal

is to profit from the differences in price movements between the two stocks, rather than concentrate on the direction of the market as a whole (Gurrib, 2014). There has been a great deal of research done on pairs trading, with many studies showing its value in lowering market risk as well as its potential for profit. (Avellaneda Lee, 2010; Vidyamurthy, 2004; Gatev et al., 2006). According to Puspaningrum (2012), the pairs trading strategy makes use of opportunity for arbitrage that arises from temporary differences in linked assets' prices that are in long-term equilibrium. If this occurs, the price of one asset will be higher than the price of the other. We then invest in a pair of stocks where the undervalued stock is purchased (going long) and the overvalued stock is sold (going short). The trade is consumated by taking opposite positions of the trade once the prices of these assets have returned to their long-term correlation. The short-term price fluctuations of the two assets generates a profit opportunity.

1.2.1 Examples of Pairs Trading in Stock Markets

One of the hardest aspects of pairs trading is figuring out which pairs have a relationship that can be regarded as fundamental and which are just coincidental. To identify profitable trading pairs, Tayal (2021) presents an algorithm to evaluate the search space and suggests an unsupervised clustering machine learning method for building the search space for pair selection. The study finds that they ultimately select better pairings, which reduce investor risk because they are less volatile and exposed to the market. Ghatak (2011) investigates the use of stock pair trading in the Indian market. The study's goal was to look into the univariate and multivariate variations of the traditional pairs trading approach. The trading algorithm is applied to the analyzed data to fully illustrate and evaluate the framework for the Indian stock market. By using trading rules to analyze daily observations of 26 Indian stock market assets, the method's performance was assessed in terms of risk and return using a database that covered the years 2008 to 2010. The results of the study demonstrate that the pair trading strategy yields a significantly higher return than naive investing. Researchers Andersen and Tronvoll (2018) examine how machine learning techniques can be used to identify statistical arbitrage opportunities in Norway’s stock market through use of pair trading strategies. Multiple supervised learning algorithms are used to predict trading signals, while clustering techniques are used to identify pairs. Returns, the Sharpe Ratio, and other model performance metrics are used as the basis for comparative analysis.

1.2.2 Pairs Trading in the Futures Market

Leung and Yan (2018) investigate the issue of trading a pair of futures contracts in a dynamic manner. They examine a mean-reverting model that uses two-factors in which the stochastic equilibrium is the center of the spot price's tendency to evolve. By resolving a utility maximization problem, they ascertain the futures price dynamics and the best futures trading approach. Through the examination of the corresponding Hamilton-Jacobi-Bellman (HJB) equation, they arrive at an explicit solution for the utility maximization problem and offer the best trading strategies. Their approaches are used in volatility futures trading (VIX) , and several numerical examples are provided to demonstrate them. In order to create a long-term, market-neutral investing strategy for pairs trading across various futures markets, Baek et al. (2020) investigate the application of machine learning techniques. Using SVM algorithm based on the dual Engle–Granger method and ECM (Error Correction Model) framework, cointegrated pairs are identified. Next, they calculate a hedge ratio. The study demonstrates that normal contango and backwardation do not always manifest in futures markets and that an algorithmic strategy for pairs trading performs well given the unique dominant price trends of each futures market. When compared to an appropriate benchmark, the pairs trading strategy produces higher risk-adjusted returns while reducing exposure to market risk across multiple futures markets. They conclude that the pairs trading strategy can be profitable and can serve as a safety net against unforeseen systemic events like the COVID-19 pandemic, according to the backtested results.

1.2.3 Pairs Trading in the Crypto-Currency Market

Lintilhac and Tourin (2017) propose a model of the optimal dynamic pairs trading strategy for a portfolio of co-integrated assets. They use stochastic control techniques to analytically calculate the optimal portfolio weightings and relate their results to several other commonly used strategies, including the static double-threshold strategy. Finally, they run an out-of-sample test and apply their model to a bitcoin portfolio using historical data from three exchanges and two cointegrating relations. While pair trading strategies have proven effective in the stock market, Fil and Kristoufek (2020) argue that because cryptocurrencies are generally perceived as unpredictable and inefficient, their performance in this domain is still unknown. More precisely, they use the distance and co-integration methods at 5-minute, 1-hour, and daily frequencies, using a basket of 26 liquid cryptocurrencies traded on the Binance exchange. The strategies do

not outperform traditional benchmarks in their backtests. Moreover, the results are very sensitive to variations in the values of the parameters and external factors, such as execution windows or transaction fees. The most popular daily distance method returns − 0.07% monthly, while higher-frequency trading performs noticeably better, returning 11.61% monthly at 5-minute frequency. Furthermore, they discover evidence of basic mean-reverting behavior in intraday prices, which is absent from daily data and lends credence to the notion that cryptocurrency markets are inefficient.

	Daily			Hourly			5-Minute			Market
	Dist.	Coint.	Random	Dist.	Coint.	Random	Dist.	Coint.	Random	BTC
Monthly profit	-0.07%	1.36%	-0.59%	3.10%	1.11%	1.97%	11.61%	4.16%	6.54%	0.43%
Annualized Sharpe	0.034	1.1	-0.62	4.1	0.66	2.4	22	12	12	0.19
Monthly number of trades	0.475	0.575	0.354	3.53	4.64	2.45	14.8	18.2	9.42	None
Roundtrip trades	23.65%	25.79%	14.37%	28.33%	38.68%	20.76%	37.15%	43.19%	25.68%	None
Length of position (days)	33.3	32.1	35.1	5.6	4.8	5.8	1.6	1.4	1.7	None
Pct of winning trades	46.06%	50.33%	33.03%	47.67%	52.73%	37.68%	53.24%	58.16%	40.99%	None
Max drawdown	26.32%	24.89%	29.48%	13.69%	15.81%	15.89%	9.02%	10.18%	10.32%	67.44%
Nominated pairs	20.0	22.2	20.0	20.0	26.1	20.0	20.0	36.3	20.0	None
Traded pairs	82.00%	87.69%	65.65%	85.42%	92.59%	68.86%	89.85%	94.74%	73.41%	None

Table 1. Performance of periodic pairs trading strategies in Binance Cryptocurrency market - Fil and Kristoufek (2020)

Thazhungal (2021) aims to investigate the potential for profiting from pair trading in the cryptocurrency market. The four most popular approaches to pairs trading—correlation analysis, distance approach, stochastic return differential approach, and cointegration analysis—form the basis of the empirical design. These approaches use the monthly closing prices of the top cryptocoins for the period spanning from January 1, 2018, to December 31, 2019. The study also conducts a simulation exercise that evaluates the payoffs and risks of a long-short strategy versus a long-only portfolio strategy. The study comes to the conclusion that in order to determine whether pairs trading can produce profits that are potentially market-neutral, one must look at the inverse relationship between the correlation coefficient and distance between different cryptocurrency pairs. Moreover, a substantial amount of evidence for the strategy's ongoing profitability comes from pairs trading simulations. Long-short portfolio strategies have been demonstrated to provide positive cumulative returns in most subsample periods in the cryptocurrency market, consistently outperforming cautious long-only portfolio strategies. An additional empirical challenge to the cryptocurrency market's market efficiency comes from the profitability of pairs trading. He does, however, draw attention to the possibility that other

elements, like implied volatility and spectral correlations, could also have a role in influencing the possible gains from pairs trading.

1.2.4 Pairs Trading in the Hedge Fund Market

An analytical framework for an investment strategy involving pairs trading is presented by Elliott et al. (2005). For the spread seen in Gaussian noise, they suggest a Gaussian Markov chain model that is mean-reverting. The calibrated model makes predictions, which are compared with later spread observations to determine the optimal investment strategy. They draw attention to the fact that any quantity in financial markets that is deemed to be out of balance could be used to generate wealth using this methodology.

1.2.5 Trading pairs in the commodity market

Using the cointegration method, Ungever (2018) looks into pairs trading strategies in the top ten agricultural futures markets. Two steps make up the pairings trading strategy: the formation stage and the trading stage, which makes use of daily futures data from 2004 to 2018. It is assumed that the cointegration error holds true for the trading period in the same way that it did for the formation period following the construction of the latter. The pairs trading strategy consists of long positions in cotton and short positions in coffee, as well as long positions in cotton and short positions in live cattle. It is noted that this strategy was profitable during both the formation and trading phases.

1.3 Alternative Trading Strategies to Pairs Trading

To choose their investments, traders and investors use a range of trading strategies in addition to pairs trading. Among them is "Buy and Hold," a long-term investment approach where investors buy stocks and hold them in the expectation that they will increase in value over time (Siegel, 2014). By purchasing and selling stocks in a single trading day, investors who employ "day trading," a short-term trading strategy, hope to profit from intraday price fluctuations (Feldman, 2019). Etzkorn (2019) defines "swing trading" as a trading strategy that involves holding a stock for a short time, usually a few days to a few weeks, with the intention of profiting from sharp price fluctuations. According to Lynch Rothchild (2000), "growth investing" is a long-term investment strategy in which investors choose companies with substantial growth potential and make investments in them with the hope that these companies will see increases in earnings and stock prices. A short-term trading strategy known as "momentum trading," according to

Moskwitz (2010), entails trading options contracts, which give traders the right to buy or sell a stock at a specific price at a later date. "Options trading" is the practice of trading futures contracts, which give traders the right to buy or sell a stock at a specific price at a later time (Kolb, 2010). Bullish, bearish, and stagnant markets are just a few of the scenarios in which options trading can be profitable (Cottle and Murray Block, 2011).

1.4 Deep Learning Methods for Correlated Stock Pairs Identification

Prospective developments in deep learning techniques have shown promise for locating related stocks for pair trading. Complex patterns and nonlinear correlations in data can be automatically found by deep learning algorithms, such as deep neural networks and deep clustering models (Chan, 2013, Araujo, Neves, and Pereira, 2021). According to Araujo, Neves, and Pereira (2021), deep learning clustering algorithms may be applied to financial data in order to uncover hidden relationships and improve the accuracy of stock matching in pairs trading strategies.

1.4.1 Autoencoders

The ability of autoencoders, a subclass of neural networks, to acquire effective data representations has garnered significant attention recently (Bengio, Lamblin, Popovici, and Larochelle, 2007). Autoencoders are an important part of the stock selection process in the research "Correlated Stock Identification in Pairs Trading Using Deep Learning Clustering" (Bengio et al., 2007). According to Vincent, Larochelle, Lajoie, Bengio, and Manzagol (2010), they use an encoder-decoder architecture to compress input data and then restore it into a lower-dimensional latent space. Vincent et al. (2010) state that finding pertinent features and patterns in the data reduces the reconstruction error. One benefit is that autoencoders can learn low-dimensional representations of complex input. According to Bengio et al. (2007), this trait is especially helpful for locating correlated stocks when trading in pairs.

1.4.2 Self Organising Maps (SOMs)

A popular kind of artificial neural network is known as a self-organizing map (SOM), which is able to arrange high-dimensional input into a lower-dimensional map. SOMs preserve topological linkages while producing a discrete representation of the input space through a competitive learning process. The ability to visualize complex data and identify patterns or

clusters is one of SOMs' advantages. Because of this characteristic, they can be good candidates for locating correlated stocks for pair trading (Bengio et al., 2007).

1.4.3 Random Forest

Random Forest is a powerful machine learning algorithm that is frequently used in a variety of fields because of its adaptability and efficiency (Jiang et al., 2020). As an example of an ensemble learning model, Random Forest builds multiple decision trees during training and combines their predictions to obtain better results (Jiang et al., 2020). This algorithm is especially skilled at managing high-dimensional datasets with a lot of features, which makes it appropriate for a variety of real-world applications. Specifically, Random Forest uses bootstrap sampling and random feature selection to reduce the likelihood of overfitting and it provides insights into the significance of features, enabling the interpretation of the input variables' relevance in prediction outcomes. Sardosky (2021) hypothesizes that random forest and tree bagging are easy to understand and estimate and are useful methods for forecasting the stock price direction of clean energy stocks. He uses random forests to predict the stock price direction of clean energy exchange traded funds. He finds that random forests and tree bagging predictions of stock prices are more accurate than those obtained from logit models. For a 20-day forecast horizon, random forest and tree bagging methods produce accuracy rates of between 85% and 90% while logit models produce accuracy rates of between 55% and 60%.

1.4.4 Support Vector Machines

Support Vector Machines (SVM) is a powerful supervised learning algorithm that excels in both classification and regression tasks (Zhang et al., 2020). They point out that this algorithm works by finding the hyperplane that best separates data points into different classes while maximizing the margin between them. SVMs are known for their ability to handle complex data distributions and nonlinear relationships through the use of kernel functions. By transforming the input space into a higher-dimensional feature space, SVMs can effectively classify data that may not be linearly separable in the original space (Zhang et al., 2020). Additionally, SVMs are robust to overfitting, making them suitable for small datasets with a high number of features. Yu (2022) notes that to optimize the performance of pairs trading strategy, accurate methods for price prediction should be employed and the support vector machine (SVM) is a typical one. The

research focuses on the China stock market, demonstrating how the SVM classifiers assist pairs trading strategy. The time period of data is from January 2020 to July 2022.

1.4.5 Gradient Boosting

Gradient boosting is a machine learning technique that generates an ensemble of weak prediction models, usually decision trees, as a prediction model for regression and classification problems. It builds the model step-by-step, just like other boosting techniques, and expands their applicability by enabling the optimization of any differentiable loss function. Leo Breiman's (2003) observation that gradient boosting can be understood as an optimization algorithm on an appropriate cost function served as the impetus for the concept of gradient boosting. Concurrently, Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean (1999) developed a more general functional gradient boosting perspective that led to explicit regression gradient boosting algorithms, i.e., algorithms that optimize a cost function over function space by iteratively selecting a function (weak hypothesis) that points in the negative gradient direction. Friedman developed these algorithms at the same time as Llew Mason. In many fields of machine learning and statistics other than regression and classification, boosting algorithms have been developed as a result of this understanding of functional gradient boosting.

1.4.6 Extreme Gradient Boosting (XGBoost)

Empirical evidence has demonstrated that tree boosting is a very successful predictive modeling technique. An open-source software library called XGBoost (eXtreme Gradient Boosting) offers a cutting-edge framework in a variety of languages, including Java, C++, Python. (Chen & Guestrin, 2016). Its objective is to offer a "Distributed, Scalable, and Portable Gradient Boosting Library." It has gained a lot of popularity recently, as demonstrated by numerous machine learning competition winning teams that view it as their preferred algorithm. The ways in which XGBoost differs from the more conventional ensemble methods are examined by Anju and Hazarika (2017). They talk about the regularization strategies that these approaches provide and how they affect the models. They also make an effort to address the question of why XGBoost appears to be such a powerful and adaptable predictive modeling technique. The main claim is that the model's local neighborhoods can be seen to be adaptively determined by tree boosting. Thus, it is possible to see how tree boosting accounts for the bias-variance tradeoff when fitting

models. XGBoost incorporates additional enhancements that enable it to manage the trade-off between bias and variance.

For numerous practical prediction tasks, the scalable tree boosting algorithm XGBoost has shown promising results, particularly when applied to tabular datasets. XGBoost, an acronym for xGBoost, is an ensemble learning approach that combines boosting techniques with decision tree power (Chen & Guestrin, 2016). This framework's main concept is to combine predictions from several weak learners—usually represented as decision trees—to produce a strong predictive model. This iterative approach facilitates the correction of errors made by preceding models in subsequent stages, thus gradually enhancing the model's accuracy. (Ahn et al, 2023). Notably, XGBoost incorporates regularization techniques, such as controlling tree depth and weights, to prevent overfitting, and employs a gradient descent optimization strategy to determine the optimal weights for these weak learners, thus addressing the shortcomings of conventional stock pairs selection algorithms. XGBoost inherits the fundamental principles of gradient boosting while introducing its own refinements for improved predictive performance. (Ahn et al, 2023, Chen & Guestrin, 2016). Its scalability, efficacy, and capacity to handle intricate data relationships have led to a notable surge in its usage. Regression and classification tasks are easily handled by XGBoost, which has been applied successfully in many domains..

1.5 Problem Statement

Pairs trading has the potential to reduce risk and generate profits, but a crucial first step in the process is determining which stocks are correlated. The presence of a strong correlation between two stocks indicates a potential trading opportunity, where deviations from this correlation are expected to revert to the mean (Avellaneda Lee, 2010). Traditional techniques for stock correlation analysis, like statistical and deep learning methods, have been extensively applied and have yielded insightful information about stock relationships. However, they may have limitations in capturing complex nonlinear dependencies and detecting subtle correlations in large and noisy financial datasets. Advanced techniques that can more accurately capture these complex relationships are becoming more and more necessary as financial markets get bigger and more complex (Araujo, Neves, Pereira, 2021).

Recent developments in machine learning, particularly deep learning, hold promise for improving stock pairing accuracy for pairs trading. These advancements present an opportunity

to raise the accuracy of stock matching. Although traditional statistical and deep learning techniques have been used previously, the release of the sophisticated machine learning algorithm XGBoost presents a viable path to improve the precision of correlated stock pair selection. However, there is a notable gap in academic research regarding the use of ensemble machine learning algorithms, specifically XGBoost for stocks selection in pairs trading. A thorough evaluation of XGBoost's effectiveness in contrast to conventional techniques is necessary, considering the stock market's growing complexity and volatility. In order to close this gap, this research effort compared conventional statistical and deep learning approaches for stock selection in pairs trading with XGBoost, a potent machine learning algorithm. The objective was to ascertain whether XGBoost outperforms other deep learning techniques and conventional statistical methods in pairs trading when it comes to stock selection. The study compared XGBoost algorithm's accuracy, precision, and recall to traditional statistical and deep learning methods used in the identification of correlated stock pairs. In doing so, this study offers insightful information to investors and traders who want to maximize their stock trading choices in a more competitive and dynamic financial landscape by leveraging XGBosst in stock pair selection.

1.6 Main Objective

The main objective was to determine whether XGBoost is a superior stock selection algorithm in pairs trading compared to other traditional statistical and deep learning methods.

1.6.1 Specific Objectives

I. To investigate techniques for identification of correlated pairs that are currently in use
II. To develop an analytical pairs identification method by application of extreme gradient boosting algorithm
III. To compare the performance of the developed analytical method to existing ones.

1.7 Research Questions

I. What are the techniques currently used for identification of correlated pairs?
II. Is it possible to develop an analytical pairs identification method by application of extreme gradient boosting algorithm?
III. How does the performance of the developed analytical method compare to existing ones.

1.8 Significance of the study

This study is significant because of the implications it has for the realm of stock trading and financial decision-making. As a proven strategy, pairs trading relies on the ability to recognize correlated stock pairs with high accuracy in order to minimize risks and maximize returns. Using XGBoost, a cutting-edge machine learning algorithm, is a novel way to handle the difficulties involved in choosing stocks. This study presents an opportunity to reshape the field of stock trading strategies by evaluating XGBoost's performance in relation to conventional techniques. In a financial world characterized by ever-shifting market dynamics, geopolitical events, and economic data, this research contributes directly to the toolkit of market participants seeking to make informed, data-driven investment decisions. In the end, this study's importance goes beyond academia; it touches on the fundamentals of financial decision-making and emphasizes the constant need to change and adapt in response to a financial ecosystem that is constantly undergoing change.

1.9 Scope of the study

This study will primarily focus on investigating the efficacy of using the XGBoost machine learning framework in stock pair identification in pairs trading strategies. It includes a detailed literature review, the development and evaluation of the XGBoost machine learning-based models, and the provision of insights and recommendations.

Chapter 2: Literature Review

2.1 Overview

Pairs trading strategies have been extensively studied in the literature, with researchers exploring various approaches to identify and exploit the relationship between correlated stocks. The profitability and effectiveness of pairs trading strategies in different financial markets have been the subject of several studies (Avellaneda & Lee, 2010; Gatev, Goetzmann, & Rouwenhorst, 2006). These studies have provided valuable insight into the potential benefits of pairs trading as a strategy for investing in markets..

2.2 Theoretical review

The pairs-trading algorithms that are currently in use in the literature, according to Deshpande and Barmish (2016), have rather strict requirements regarding the spread function and the underlying stochastic stock-price processes. They analyze a pairs trading algorithm that operates under less constraining assumptions than the previous algorithms. Control theory forms the foundation of their viewpoint. In a feedback context, they present a general pairs-trading algorithm that lets the user specify any spread function and dynamically adjusts investment levels over time. This function views the stocks as a tradeable pair when it and the price process satisfy a specific mean-reversion condition. They demonstrate how their trading algorithm, which draws inspiration from the control theorem, produces positive expected growth in account value. In their final section, they test their algorithm using stock price pairs fitted to a widely-used spread function in the literature and historical trading data. The simulation results for these tests indicate strong growth with very small drawbacks.

When trading in pairs, one stock is held in a long position and another in a short position. Closing the pairs position aims to determine the best moment to sell the long position and buy back the short position. Liu et al. (2023) present an optimal pairs-trading selling rule with trading constraints. A two-state Markov chain with the states "trading allowed" and "trading not allowed" describes the trading permission process, while the underlying stock prices show a two-dimensional geometric Brownian motion. They demonstrate how a threshold curve, which is produced by resolving the related Hamilton-Jacobi-Bellman (HJB) equations (quasi-variational inequalities), can be used to identify the best course of action. They yield an

analytical solution as well as a verification method. They report on their numerical experiments to demonstrate optimal policies and value functions to guide in pairs trading.

Using daily data from 1962 to 2002, Gatev and Goetzmann (2002) test the pairs trading strategy. Stocks are paired according to how close their normalized historical price differences are to one another. Self-financing pairs portfolios can achieve average annual alpha returns of up to 11% by using a straightforward trading rule. Usually, the profits outpace even the most cautious estimates of transaction costs. The "pairs" effect appears to be different from previously reported reversal profits, according to bootstrap results. The robustness of the alpha returns implies that trading in pairs profited from the transient mispricing of closely related stocks. Unlike traditional risk measures, they relate the return to the existence of a correlation. Hong and Hwang (2022) examine whether the use of firm characteristics to identify pairs results in fewer spurious pairs being produced by multiple hypothesis testing. The results show that the portfolios of pairs with more similar firm characteristics outperform those that are fundamentally less similar by lowering the non-convergence risk. Although empirical results indicate that pairs trading profitability has declined since 2003 and is no longer significant, the cost of trading spurious pairs is still substantial.

2.2.1 Statistical Approaches to Stock Pair Identification

According to Jirapongpan and Phumchusri (2020), numerous scholars are attempting to investigate and develop mathematical models to enhance the pairs trading approach on a variety of assets, including forex, stocks, cryptocurrency, and financial derivatives (options, forwards, futures, and swaps). These mathematical models include machine learning, Kalmann filter, cointegration method, and Ordinary Least Squares (OLS). The models aim to produce accurate signals from asset pairs in order to optimize the return through the use of statistical arbitrage of pairs trading strategy. Pairs trading and its derivatives are statistical arbitrage strategies that depend on the predictable construction of mean-reverting spreads.

2.2.2 Pearson Correlation

According to Lo, Mamaysky, and Wang (2000), the Pearson correlation coefficient is a statistical measure that is frequently used to quantify the linear relationship between two variables. The development of pairs trading strategies has benefited greatly from these techniques, which have

also formed the basis for stock correlation analysis. According to Hendry, Banerjee, Dolado, Galbraith (1993) and Vidyamurthy (2004), the cointegration technique is a more effective way to extract profit potential from transient pricing anomalies between two stock prices driven by common underlying factors than the correlation criterion. This aids in identifying the two stocks that make a pair. Wang (2009) assesses the efficacy of two distinct correlation measures in initiating trades in pairs trading scenarios involving high-frequency stock prices. The widely used Pearson correlation is one type of correlation measure, while the robust Maronna correlation is another. The three trigger mechanisms—which are used to initiate trades—and their respective properties are defined by means of these correlation measures. Using historical data, the researcher compares the performance of trading strategies with the three different types of triggering mechanisms. Based on the findings, statistical tests are conducted. He discovers that trading strategies built around trigger mechanisms that use strong correlation measures consistently produce lower returns but better risk characteristics.

2.2.3 Cointegration

In the past, correlated stocks in pairs trading have been found using statistical measures like cointegration or distance metrics. When stock pairs cointegrate, we can identify a long-term relationship that can be profitablely manipulated. This kind of situation usually leads to the discovery of stock pairs with high correlation through the use of cointegration tests such as the Granger-Engle and Johansen tests (Engel and Granger (1987)). The cointegration methodology is the main focus of Puspaningrum (2012). The researcher analyzes buy and sell investment strategies involving more than two stocks using cointegration analysis and mean rversion concepts. If the portfolio is known to fluctuate around its equilibrium value, then any deviations from that value can be traded against. The cointegration coefficients weighted (CCW) rule serves as the foundation for the researcher's pairs trading approach. The process of maximizing the minimum total profit (MTP) over a designated trading horizon yields the ideal pre-set boundary value. The number of trades made during the trading horizon and the minimum profit per trade determine the MTP. Cointegration tests are used by Caldeira and Moura (2013) to determine which stocks to use in pairs trading strategies. They not only model the residuals and estimate the long-term equilibrium, but they also select the stock pairs to include in a pairs trading portfolio based on an in-sample profitability indicator. Data from January 2005 to October 2012,

which is available from the Sao Paulo stock exchange, is used to evaluate the strategy's profitability. According to their empirical analysis, the suggested strategy has a low beta with the market, a Sharpe Ratio of 1.34, and excess returns of 16.38% annually.

Afawubo and Huck (2015) investigate the effectiveness of a pairs trading strategy based on different pairs identification techniques using data from the S&P 500 index. They draw attention to the fact that although the distance method has numerous empirical applications in the literature, it can also be used in conjunction with other well-known statistical and econometric methods like cointegration and stationarity. However, using these techniques has the drawback of increasing the computational demands placed on the trading system. When stocks diverge from their equilibrium, trades are opened. Their results confirm that, once risk and transaction costs are taken into account, the distance method yields very small excess returns. Furthermore, they demonstrate that pairing selection that meets the stationarity criterion performs poorly, whereas cointegration produces a high, stable, and robust return. Huck (2015) continues to investigate the performance of a pairs trading system based on different pairs selection techniques (distance, stationarity, cointegration) over a 10-year period using data from the S&P 500 and the Nikkei 225. Cointegration appears more effective and better from a classical framework on both markets. Pairs trading strategies fared exceptionally well on the American market during the 2008 financial crisis and, to a lesser extent, in Japan as well. A high VIX index, also known as the "investor fear gauge," is linked to bearish times. The researcher investigates the relationship between volatility/VIX timing and pairs trading performance using a modified trading system. Volatility Timing has no economic value in the context of pairs trading after it was determined that cointegration is the best selection strategy.

Jaeger (2016) uses a cointegration pairs trading algorithm developed by Gatev et al. (2006) using data from the S&P 500 Index. The investigator finds that the pairs trading strategy has lost money over time and notes that pairing with cointegration or using the cointegrating regression as the basis for the trading rule in the Gatev et al. (2006) algorithm does not seem to be able to improve the recent bad performance. A new pairs trading strategy is developed, which essentially involves updating the equilibrium model dynamically to account for changes in the fundamental equilibrium. It is suggested that pairs in a portfolio be chosen using a trend-following approach, with funds allocated to pairs exhibiting the best recent backtest

performance. In the research, the strategy yields an average annual excess return of 17% in the test period (1985 to 2010) but returns decrease significantly in the validation period (post 2010).

Brunetti and Luca (2020) compare the overall profitability of a cointegration-based pairs trading strategy when stock pairs are pre-selected based on seven different metrics. They discover that pre-selection is important because, depending on the metrics employed, the excess returns differ significantly in terms of average and variability. Even after taking into consideration commissions and cut rules, market impact, a stricter definition of the Spread reversion to the equilibrium, and different cointegration tests, there are still differences in profitability by pre-selection metrics. Furthermore, it is discovered that the profitability of pairs trading varies with respect to exposure to systematic stock-market risk factors, depending on the pre-selection metric employed. The diffusion limit of first-order vector autoregressive (VAR(1)) cointegration models is often taken into consideration when developing trading rules for continuous-time pairs. The empirical identification of cointegration effects typically makes use of discrete-time error correction representations of vector autoregressive (VAR(p)) processes, since this enables delayed price deviation adjustment. This motivates Yan and Wong (2022) to study the dynamic pairs trading problem in continuous time under a class of path-dependent models. They demonstrate the existence of the best course of action and how it relates to a system of Riccati partial differential equations under specific regular conditions. Functional Itô's calculus is used to develop the proof. They conduct a numerical analysis to look at how sensitive the pairs trading strategy is to the initial market conditions and history.

2.2.4 Time Series Analysis

A popular statistical method in finance for examining past data and spotting trends in asset prices over time is time series analysis. Time series analysis can reveal trends, seasonal patterns, and cyclical fluctuations, among other insightful information about the behavior of asset prices (Brockwell & Davis, 2016). Moreover, time series analysis can be used to develop trading strategies, including pairs trading, based on historical data (Avellaneda & Lee, 2010). Time series analysis has been investigated in several studies as a potential tool for creating pairs trading strategies. For instance, Chen and Li (2017) discovered that a pairs trading strategy they created using vector autoregression (VAR) models outperformed the benchmark index. Similarly, Zhang and Li (2018) developed a pairs trading strategy in the Chinese stock market and made

significant profits using VAR models and cointegration analysis. Alternative time series analysis methods, like wavelet analysis and machine learning algorithms, have been investigated in other studies in order to create pairs trading strategies. For instance, Ma and Zhang (2019) created a pairs trading strategy that produced notable profits by using wavelet analysis to find stock pairs with a high degree of correlation.

2.2.5 Deep Learning Approaches to Stock Pair Identification

The field of stock correlation analysis now has more options thanks to the development of deep learning clustering techniques. Araújo, Neves, and Pereira (2021) have noted that deep learning models, like deep neural networks and deep clustering algorithms, have demonstrated potential in automatically identifying complex patterns and nonlinear relationships in data. Autoencoders and self-organizing maps are two deep learning models that show promise in revealing complex relationships and patterns in stock price data (Araujo et al., 2021). These models attempt to deal with the intricacies of non-linear relationships, advancing the frameworks of identifying correlated stock pairs with greater accuracy.

2.2.6 Autoencoders

The capacity of autoencoders, a class of neural networks, to acquire effective data representations has drawn a lot of attention lately (Bengio, Lamblin, Popovici, & Larochelle, 2007). They are made up of a decoder network that reconstructs the original input from the latent representation and an encoder network that compresses the input data into a lower-dimensional representation (latent space). The preprocessed stock data is used to train an autoencoder, which helps the model identify underlying patterns and relationships among the stocks. After that, the latent space can be utilized for clustering analysis, in which cases comparable stocks should have comparable representations. The research paper "Correlated Stock Identification in Pairs Trading Using Deep Learning Clustering" (Bengio et al., 2007) highlights the significance of autoencoders.

2.2.7 Self-Organizing Maps (SOMs)

SOMs are a type of neural network that can project high-dimensional data onto a lower-dimensional grid while preserving the topological structure of the data. (Kohonen, 1990).

In the context of stock clustering, SOMs can be used to map the stocks onto a grid where neighboring grid cells represent similar stocks. The SOM algorithm iteratively adjusts the grid representation to capture the underlying patterns in the stock data. The resulting SOM can then be analyzed to identify clusters of stocks based on their correlations or similarities. (Kohonen, 1995). The core principle behind SOMs is competitive learning, where neurons within a grid-like topology compete to represent different regions of the input space. During training, each neuron adjusts its weights to become more similar to a subset of the input data, effectively learning to represent a specific feature or pattern present in the data. The competitive learning process is complemented by a neighborhood function that determines the extent to which neighboring neurons update their weights, allowing for the preservation of the input space's topological relationships. (Kohonen, 1995).

SOMs offer several advantages, including their ability to uncover hidden structures in data without requiring explicit supervision, making them particularly useful for exploratory data analysis and visualization. Additionally, SOMs are robust to noise and can handle high-dimensional data efficiently, facilitating the identification of meaningful patterns in complex datasets. (Kohonen, 2004). Despite their strengths, SOMs also present some limitations and challenges. The choice of parameters, such as the grid size and learning rate schedule, can significantly impact the quality of the resulting map. Furthermore, interpreting the learned representations from SOMs may require additional post-processing and analysis, as the mapping between input features and neuron activations may not always be straightforward. (Kohonen, 2004).

2.2.8 Support Vector Machines

Support Vector Machines (SVMs) represent a class of supervised machine learning models utilized extensively across various domains, ranging from bioinformatics to image recognition. SVMs excel in scenarios characterized by high-dimensional feature spaces, owing to their capability to discern complex patterns and extract meaningful decision boundaries. As articulated by Vapnik (1995), the fundamental objective of SVMs is to identify the hyperplane that optimally segregates classes in the feature space, with the intent of maximizing the margin – the distance between the hyperplane and the nearest data points from each class, termed support vectors. This margin maximization principle underscores the robustness of SVMs in

classification tasks, enhancing their ability to generalize to unseen data. The kernel trick stands out as a pivotal aspect of SVMs, enabling them to undertake nonlinear classification effectively. By transforming the input data into a higher-dimensional space, SVMs can achieve linear separability, thereby extending their applicability to diverse datasets. This concept of kernelization, elucidated by Schölkopf et al. (2001), underscores SVMs' flexibility in capturing intricate relationships within the data, thus enhancing their predictive performance. Moreover, SVMs exhibit a notable property of regularization, governed by the parameter C, which balances the trade-off between model complexity and classification accuracy. This regularization mechanism, as expounded by Cortes and Vapnik (1995), endows SVMs with the ability to resist overfitting, thereby promoting generalization on unseen data.

The optimization framework underpinning SVMs involves solving a quadratic programming problem, aimed at identifying the optimal hyperplane that maximizes the margin while minimizing classification errors. Various optimization techniques, such as Sequential Minimal Optimization (SMO), have been devised to expedite this process, facilitating efficient model training (Platt, 1998). While SVMs offer several advantages, including effectiveness in high-dimensional spaces and robustness against overfitting, they also entail certain limitations, such as computational complexity and sensitivity to kernel selection (Hsu et al., 2003). Huang et al. (2017) evaluate the classification accuracy, ROC, F-measure, and computational times of training SVM and evaluate the prediction performance of SVM and SVM ensembles over small and large-scale breast cancer datasets. They point out that several statistical and machine-learning methods have been used to create different models for predicting breast cancer. They demonstrate that Support vector machines (SVM) perform better than numerous competing techniques. They however note that the kernel function must be chosen before building the SVM classifier. The prediction performances of SVM based on various kernel functions, however, have yet to be the subject of many studies. Furthermore, it is unclear if SVM classifier ensembles, which have been suggested as a way to enhance the performance of individual classifiers, can predict better than individual SVM classifiers.

2.2.9 Random Forests

According to Breiman (2001), random forests are made up of a combination of tree predictors where each tree in the forest is dependent on the values of a random vector that is independently

sampled and has the same distribution for all the trees in the forest. As the number of trees in a forest increases, the generalization error converges to a limit. The strength of each individual tree in the forest and the correlation between them determine the generalization error of a forest of tree classifiers. Error rates obtained from randomly selecting features to split each node are more robust against noise, but still compare favorably to those of Adaboost (Y. Freund & R. Schapire, Machine Learning: Proceedings of the Thirteenth International conference, ***, 148–156). Internal estimates track correlation, error, and strength and are used to illustrate how the number of features used in the splitting changes. Variable importance is also measured using internal estimates.

Random Forests are composed of multiple decision trees, where each tree is trained on a random subset of the training data and uses a random subset of features. The predictions from individual trees are then aggregated to make the final prediction, either through averaging (for regression) or voting (for classification). Random Forests were first introduced by Leo Breiman in 2001 and have since become widely adopted in various domains due to their robustness and versatility. (Breiman, 2001) notes that the key idea behind Random Forests is to reduce the variance of individual decision trees by introducing randomness in the training process. This randomness comes from two main sources: random sampling of the training data and random feature selection. By training each tree on a different subset of the data and considering only a subset of features at each split, Random Forests can decorrelate the trees and reduce overfitting, leading to better generalization performance. One of the strengths of Random Forests is their ability to handle high-dimensional data and datasets with a large number of features. They are also robust to noisy data and outliers, making them suitable for a wide range of real-world applications. Moreover, Random Forests provide a built-in mechanism for feature importance estimation, allowing users to gain insights into which features are most influential in making predictions. (Breiman, 2001). Despite their effectiveness, Random Forests come with some limitations. They can be computationally expensive, especially for large datasets with many trees and deep trees. Additionally, the interpretability of Random Forests can be limited compared to simpler models like decision trees, as the final prediction is based on the collective decision of multiple trees.

2.2.10 Extreme Gradient Boosting

One important area of machine learning is tree boosting. The system is designed for fast parallel tree construction and is built to withstand faults in a distributed environment. The system is built to be fault tolerant in a distributed environment and is optimized for quick parallel tree construction. With distributed computing, XGBoost can handle billions of samples in addition to tens of millions of samples on a single node. Chen and Guestrin (2016) propose a new sparsity-aware machine learning algorithm for sparse for approximate tree learning, which together led to the development of the scalable tree boosting system known as XGBoost. They also provide information on sharding, data compression, and cache access patterns. It has been widely used in computer vision, natural language processing, financial prediction, and other fields. Its efficacy, scalability, and capacity to handle intricate relationships within the data have led to a notable surge in its popularity. XGBoost excels in both classification and regression tasks. Gradient boosting of regression trees, according to Friedman (2001), yields highly robust, competitive, and interpretable procedures for both regression and classification. It is particularly suitable for mining less-than-clean data. Friedman also discusses the similarities and differences between this approach and the boosting methods of Freund and Shapire (2020).

Anju and Sharma (2018) note that the boosting technique creates a strong learner with a higher accuracy by combining several weak learners. The main purposes of boosting are to handle missing values and avoid overfitting. The accuracy of the Random Forest, XGBoost, and AdaBoost algorithms is compared in this study. They discover that XGBoost uses the fewest resources possible to solve a wide range of real-world issues. The comparative analysis shows that big data applications requiring parallel computations are best suited for XGBoost, which has the highest accuracy. XGBoost is a combination of regularization and gradient boosting techniques that provides robustness against overfitting and high predictive accuracy (Chen and Guestrin, 2016). The algorithm uses an improved tree-boosting framework (XGBoost (eXtreme Gradient Boosting), 2023) that combines parallel computing and tree pruning to achieve remarkable results. Numerous studies have demonstrated the effectiveness of XGBoost, a useful tool in the machine learning toolbox, in improving the performance of tasks related to classification, regression, and ranking.

2.3 Research Gap

Pairs trading has historically relied on traditional statistical and deep learning techniques. These techniques seek to determine whether stock price correlations are linear or non-linear (Avellaneda Lee, (2010)); Araujo, Neves, and Pereira, (2021)). However, according to Li and Li (2018), these traditional methods are not sufficient to address complex non-linear dependencies between stocks. Even if the relationships are not strictly linear, it is important to find stocks that exhibit strong correlation patterns when trading in pairs (Kumar and Avinash, 2021). Such relationships might be missed by traditional statistical methods, leading to the selection of subpar trading pairs and results. In particular, the deep learning techniques are limited by their role as incomprehensible "black boxes" (Araujo et al., 2021). Due to the lack of transparency, it is challenging to determine the underlying causes of the detected stock clusters and to make wise trading decisions. Deep learning models, according to Araujo et al. (2021), are also prone to overfitting, which can catch anomalies or noise in the training set and prevent the models from generalizing to new data. False correlations and less-than-ideal trading decisions can arise from overfitting. Deep learning models can be difficult to train because of their computational complexity and length of training. This can be an issue in high-frequency trading or real-time situations where quick decision-making is crucial (Kumar and Avinash, 2021). Lastly, the process is labor- and resource-intensive due to the intricacy of the model architecture and the requirement for hyperparameter tuning, which calls for experience and extensive experimentation (Araujo et al., 2021).

The goal of this research is to explore the use of traditional and machine learning algorithms in selection of trading pairs, and to specifically investigate whether the XGBoost machine learning algorithm perfoms better in this task compared to the other models. Our goal with XGBoost is to build robust models that can detect and capitalize on stock price divergences rapidly. This study adds to the body of research on pairs trading by examining the potential of XGBoost as a potent tool for stock pairs identification and trading. The study's findings will shed light on how well XGBoost performs in pairs trading and what that means for making financial decisions.

Chapter 3: Research Methodology

In order to achieve the research goals outlined in Chapter 1, each step that needs to be taken is broken down in this chapter.

3.1 Research design

The respected CRISP-DM (Cross-Industry Standard Process for Data Mining) model is used in the study as a method for executing data mining projects. The model provides an orderly, systematic way to handle data mining requests and makes it easier to understand the underlying data. It consists of six separate stages: Defining the project's goals and needs from a business standpoint is the first phase, known as "business understanding." The second phase, which concentrates on data collection and investigation, is called data understanding. This entails compiling the pertinent datasets, analyzing their composition and organization, and spotting any possible problems or contradictions. This stage offers insightful information that helps with the later phases of data preparation and modeling.

The third stage is data preparation, which entails preprocessing and changing the data to guarantee its accuracy and fit for modeling. In order to create a clean and structured dataset that is suitable for precise and effective modeling, this may entail tasks like data integration, feature engineering, and data cleansing. It also involves handling any missing, inconsistent, or duplicate data. This is an important step because the data mining models' performance is directly impacted by the caliber of the prepared data. The fourth step, modeling, is choosing and using suitable data mining methods on the ready data. To identify the optimal models for the particular problem and data at hand, it is necessary to experiment with different algorithms and techniques. Then, in order to maximize performance, the model parameters are changed. A variety of evaluation metrics and cross-validation techniques are employed to evaluate the models in order to guarantee that the models selected perform well when applied to unobserved data.

The models are assessed to determine how well they perform in relation to the research objectives after they have been built and improved. This entails putting the models through their paces on an independent dataset and assessing their effectiveness with pertinent assessment metrics. This is crucial in ensuring that the models meet the study's goals and provide useful trading data. Deployment, the last step, entails integrating the models into systems and business

usage. By carefully planning and executing the models' deployment, the researcher ensures that the project's objectives are fulfilled and that the insights gained from data mining are successfully applied to inform business decisions and strategies.

To sum up, the CRISP-DM model offers a thorough, organized method for handling data mining, guaranteeing that every stage of the project is carried out successfully and on schedule. By using the CRISP-DM methodology, the researcher can better understand the data, create accurate and reliable models, and ultimately offer recommendations for tactics and decisions that will help the business succeed. A summary of the steps are represented in Figure 5 below.

Fig. 1: CRISPM - DM Model - Hotz (2018)

3.2 Data sources and collection methods

Our stock pair selection strategy is based on daily price data for a sample of 20 stocks that are listed on the S&P 500 index between January 1st, 2010 and December 31st, 2022. The validation period runs from January 1st, 2023, to December 31st, 2023. The information is electronically downloaded from the Yahoo Finance website and pertains to a number of industries, including consumer goods, healthcare, energy, and technology.

AAPL	(Apple Inc.)
AMZN	(Amazon.com Inc.)
MSFT	(Microsoft Corporation)
GOOGL	(Alphabet Inc. - Class A)
JPM	(JPMorgan Chase & Co.)
META	(META Inc. - Class A)
TSLA	(Tesla Inc.)
JNJ	(Johnson & Johnson)
NVDA	(NVIDIA Corporation)
V	(Visa Inc. - Class A)
WMT	(Walmart Inc.)
DIS	(The Walt Disney Company)
BAC	(Bank of America Corporation)
MA	(Mastercard Incorporated - Class A)
HD	(The Home Depot Inc.)

PYPL	(PayPal Holdings Inc.)
PFE	(Pfizer Inc.)
NFLX	(Netflix Inc.)
VZ	(Verizon Communications Inc.)
XOM	(Exxon Mobil Corporation)

Table 2. Ticker symbols for the 20 randomly selected stocks

3.3 Exploratory Data Analysis (EDA)

Understanding the connections between various assets or securities in the financial markets is crucial to pairs trading strategies. In order to understand the dataset and underlying market dynamics, comprehensive exploratory data analysis (EDA) must be carried out prior to using machine learning techniques to find trading-suitable pairs. Using the matplotlib and pyplot libraries in Python, we looked into the price trends of the selected stocks over the course of the ten-year study period. Several data plots were visually inspected in order to identify potential outliers in the dataset. Outlier detection is crucial because outliers have the ability to skew analysis results.

3.4 Data Cleaning

This step involved taking the raw data and transforming, fixing, and eliminating errors and inconsistencies. The objective was to produce a robust, trustworthy dataset for analysis and model building. Data inspections and explorations were among the procedures and methods utilized to clean the dataset. It included examining summary statistics (e.g. the mean and standard deviation, and any missing data) to identify potential anomalies and errors (noise). We noted that for 2 stocks, there had been a name change during the 10-year period under study. Google had changed its name to Alphabet, while Facebook had changed its name to Meta. We corrected for these changes by updating the ticker names and symbols to the new names.

3.5 Data Preprocessing

Data preprocessing plays a crucial role in ensuring the quality and suitability of the dataset for machine learning analysis. Preprocessing steps typically include handling missing values, standardizing or normalizing features, and encoding categorical variables (Fernández-Delgado et al., 2014). Additionally, feature selection or dimensionality reduction techniques may be employed to reduce the computational complexity and improve the model's generalization performance (Fernández-Delgado et al., 2014). By carefully preprocessing the data, we can mitigate issues such as overfitting and improve the overall performance of the predictive models. Preprocessing techniques including scaling and transforming the data distribution were used to enhance the effectiveness of the machine learning algorithms. We used Python modules and packages for the data extraction procedure and reviewed the data frame's shape. The dataset that was utilized is broken down into 20 columns for each of the 20 stocks and 3,272 rows for each stock's daily price over the ten-year period.

3.6 Data Transformation and Feature Scaling

This step involved transforming the data into the right format for modeling. To ensure the consistency and reliability of the data, we conducted data normalization. We did this by standardizing the data, i.e., subtracting the mean of the adjusted closing price for each stock from the adjusted closing price data point and dividing by the standard deviation. To ensure the data has a common range, we further scaled the data using the MinMaxScaler function from the sklearn Python library. Varying data scales can affect the performance of machine learning algorithms hence the need for scaling. We used the sklearn module in Python to scale the features. We stored this normalized dataset in a data frame. Incompatible data types were identified and converted to compatible data types to facilitate analysis. Date and time were standardized to Pandas format for easier manipulation during analysis. Integers were also converted to float. Data transformation included categorical variable encoding. Being a classification problem, where pairs of data need to be correctly classified for use in a pairs trading strategy, we needed to encode the input ticker pairs as they could not be directly used as input in the machine learning algorithms. We encoded them to numerical values by use of the Label encoding method, which creates a numerical variable for each category.

3.7 Feature selection

This step entailed the identification of the most relevant or informative variables in the dataset. This is a crucial step to reduce high dimensionality that typically leads to overfitting. We created a list of all possible stock pairs, given the universe of 20 stocks. We then created a dictionary of stock correlations between each pair. Finally, we set a correlation threshold of 0.8 (80%), meaning we were only interested in stock pairs with correlation equal to or greater than 0.8 as input pairs into our model. To achieve this, we used the combinations function from the itertools Python library and the pairwise_distances function from the sklearn.metrics Python library. The qualifying data pairs were stored in a data frame.

3.8 Model selection and rationale

Based on the research objectives, model selection and rationale were informed by the type of problem we were trying to solve. The problem is broken down to selecting robust pairs of stocks for pairs trading using various machine learning algorithms. The hypothesis is that the XGBoost algorithm does a superior job in identifying correlated stock pairs compared to traditional stock identification methods and even other machine learning algorithms

Conceptual Framework: Stock Pairs Selection Framework

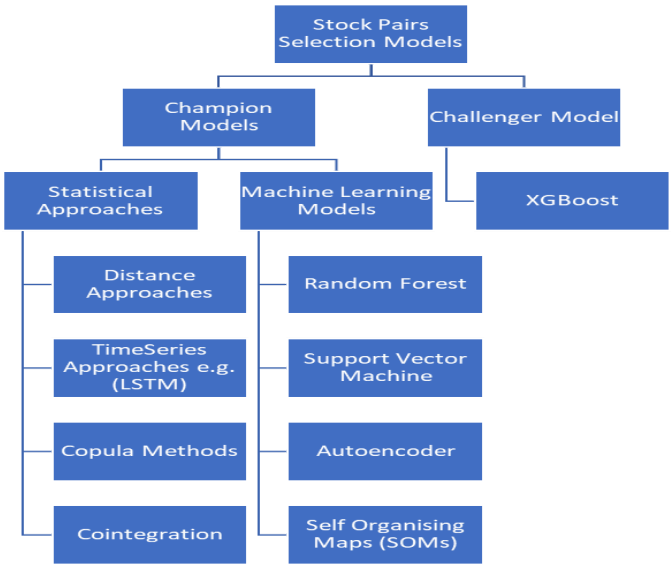


Fig. 2. Conceptual Framework: Stock Pairs Selection Framework

3.9 Modelling

With the scaled and transformed data stored in a data frame, we embarked on the modeling effort. We split the data into a train and test dataset with an 80:20 split. We then proceeded to use the different modeling algorithms to select trading pairs.

3.9.1 Distance Approach

The distance approach relies on calculating a distance metric, such as Euclidean distance or correlation distance, between pairs of securities to determine their similarity or divergence in price movements. The method calculates the distance between each pair of securities based on their historical price movements. We set a threshold value to identify pairs with sufficiently high similarity or correlation based on the calculated distances.

3.9.2 Random Forest

Several decision trees are combined in the Random Forest ensemble learning technique to generate predictions. In comparison to any single tree, it produces more accurate and consistent results by combining the predictions of individual trees. It is recognized for its strong performance in predictions and is a member of the decision tree-based algorithm family. Recursively dividing the feature space into regions and generating predictions based on the average value or majority class of samples within each region, decision trees are straightforward but effective models. They can, however, be sensitive to changes in the training set and are prone to overfitting. This unpredictability improves the model's capacity for generalization while lowering correlation between trees. Using random feature subsets and bootstrapped training data samples, a predetermined number of decision trees are grown throughout the training process. With each tree being trained separately on its own subset of data, a variety of trees with possibly distinct structures and predictive powers are produced. A majority vote or an average of the predictions made by each individual tree determines the Random Forest model's final prediction for classification tasks. The relative contribution of each feature to the predictive performance of the model is indicated by Random Forest's measure of feature importance. The ensemble averaging and randomization methods used during training make Random Forest resistant to noisy and overfitting data. It is suited for a variety of applications due to its usual high predictive accuracy and generalization performance on unknown data.

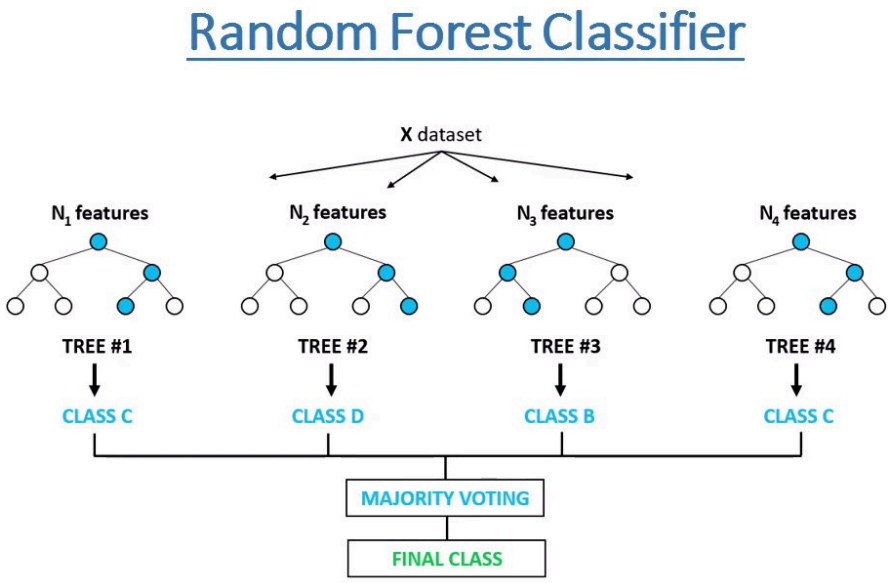


Fig. 3. Random Forest Classifier framework (Khushaktov - 2022)

We trained the Random Forest model on the training dataset, enabling it to discover the underlying patterns and relationships, by applying the RandomForestClassifier module from the Python library, sklearn.ensemble, to the train and test data. The Random Forest classifier model's feature importance scores were then examined in order to determine which features were most crucial in predicting correlated stock pairs. The relative contribution of each feature to the predictive performance of the model is indicated by feature importance scores. Then, we chose pairs of securities with high predicted correlation scores as possible candidates for pairs trading strategies. We did this by using the trained Random Forest model to predict the correlation or divergence between pairs of securities in the test dataset.

3.9.3 Support Vector Machine (SVM)

Support Vector Machine (SVM), a supervised learning algorithm, is used for tasks involving regression and classification. It is particularly effective in solving problems involving binary classification. Finding the optimal hyperplane in feature space to divide the data points into different classes is the aim of this process. The goal is to maximize the margin between the closest data points for each class and the hyperplane. The time interval that divides the closest data points, also known as support vectors, from the hyperplane is known as the margin. Because

it increases the model's ability to accurately classify new data points and boosts generalization performance, SVM seeks to maximize this margin. SVM can handle nonlinear decision boundaries by using kernel functions (like linear, polynomial, or radial basis functions) to map the input features into a higher-dimensional space. In the higher-dimensional space, the Support Vector Machine (SVM) searches for a linear hyperplane that separates the classes.

SVM is used in pairs trading strategies to find stocks that are cointegrated or have comparable price patterns by analyzing historical stock price data. The algorithm learns to classify new pairs based on their features by training an SVM model on historical price data and labels indicating whether pairs of stocks are suitable for pairs trading. SVM is flexible for a range of dataset types because it can handle both continuous and categorical features. It can handle high-dimensional feature spaces with effectiveness and is resilient to outliers. To strike a balance between the classification error and the margin's maximization, SVM adds a regularization parameter (C). While a larger C value may produce a narrower margin but fewer misclassifications, a smaller C value permits a wider margin but may result in misclassification errors. SVM uses the decision boundary (hyperplane) as a point of reference to classify data points into one of two groups when it comes to binary classification. By increasing the margin between positive and negative class instances, it gains the ability to discriminate between them.

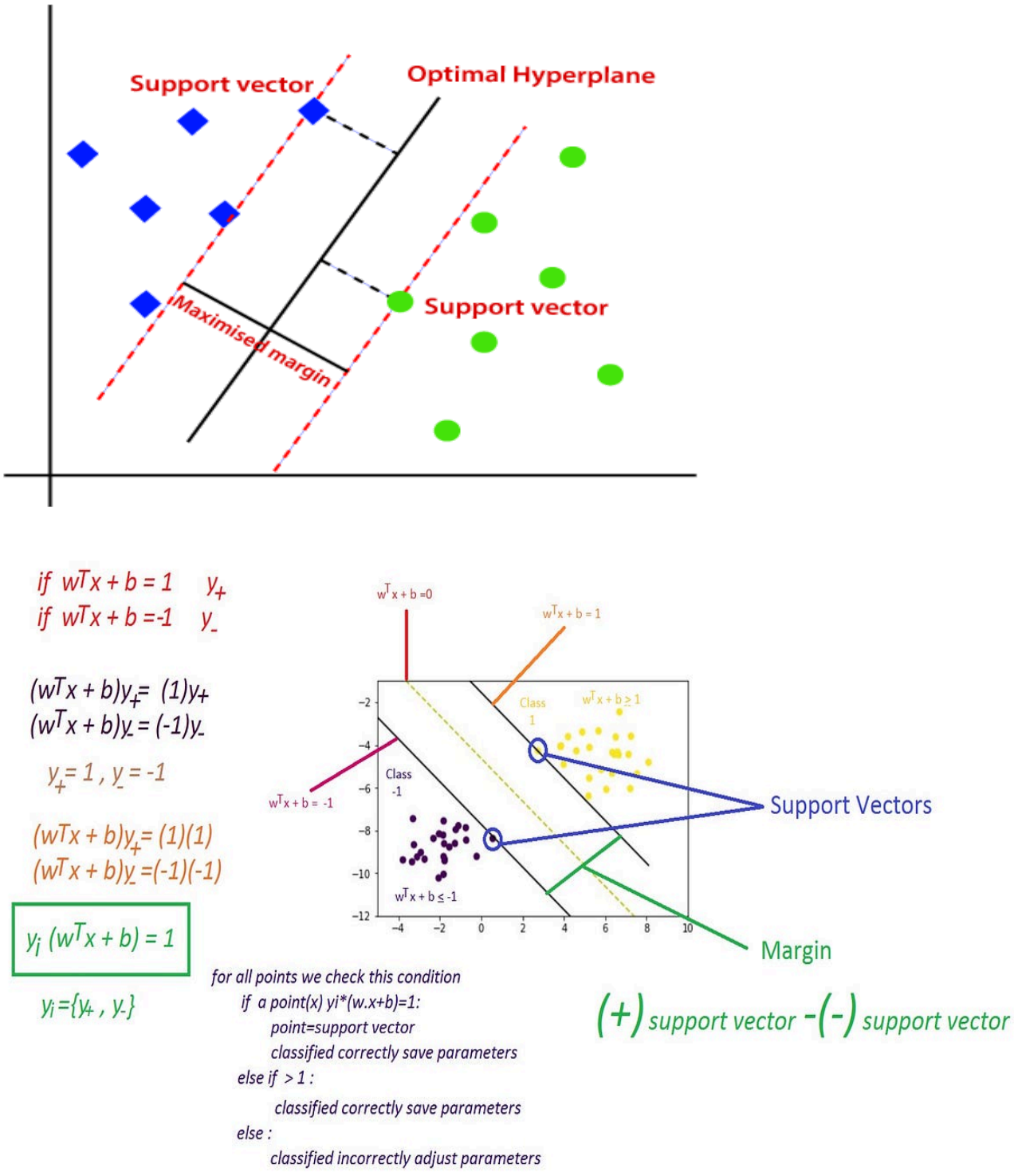


Fig. 4. Support Vector Machine Classifier framework (Pankaj - 2022)

SVM can be expanded to support multi-class classification by utilizing one-versus-one or one-versus-rest strategies. Whereas SVM creates a single hyperplane for each class against the rest in one-vs-rest, it creates multiple hyperplanes to divide each pair of classes in one-vs-one. Metrics like accuracy, precision, recall, F1-score, and the receiver operating characteristic (ROC) curve are used to assess the performance of SVMs. Cross-validation aids in evaluating the model's resilience to new data and capacity for generalization. To train the encoded train and test data, we utilized the SVC function from the Python library sklearn.svm. We assessed the model's performance by looking at its accuracy, recall, and F1-Score metrics after it had been fitted.

3.9.4 Autoencoder Model

An artificial neural network type called an autoencoder is used for unsupervised learning tasks, especially dimensionality reduction and data reconstruction. The primary goal of autoencoders is to automatically learn a compressed representation of the input data, which can be applied to a variety of tasks like anomaly detection, feature extraction, and data denoising. There are two primary parts to it: a decoder and an encoder. The encoder takes the input data and maps it to a lower-dimensional latent space representation. The decoder then takes this latent representation and attempts to reconstruct the original input data. The goal is to minimize the difference between the input and the reconstructed output, effectively learning to compress and decompress the data.

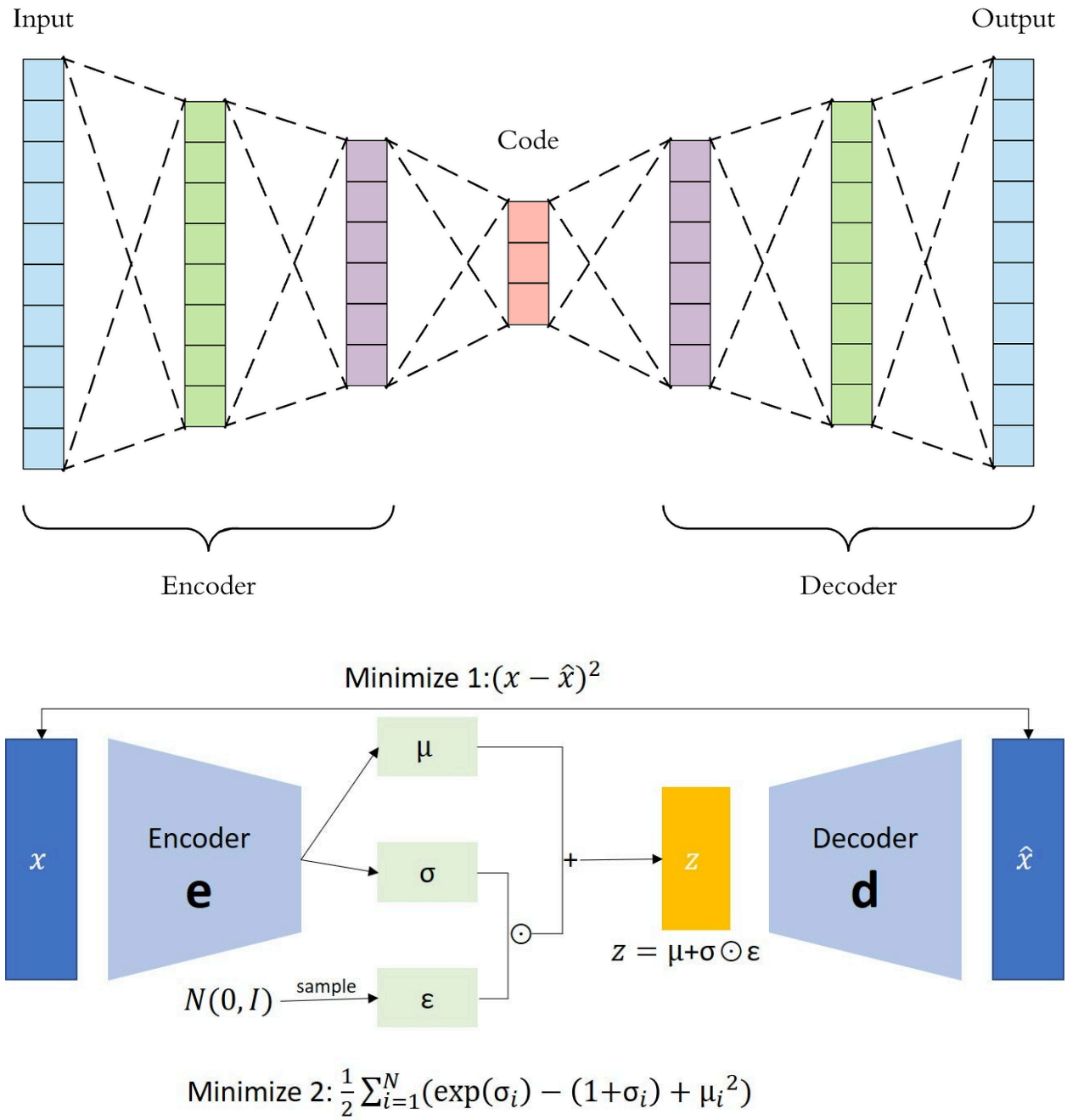


Fig. 5. Autoencoder Classifier framework (Dertat - 2023)

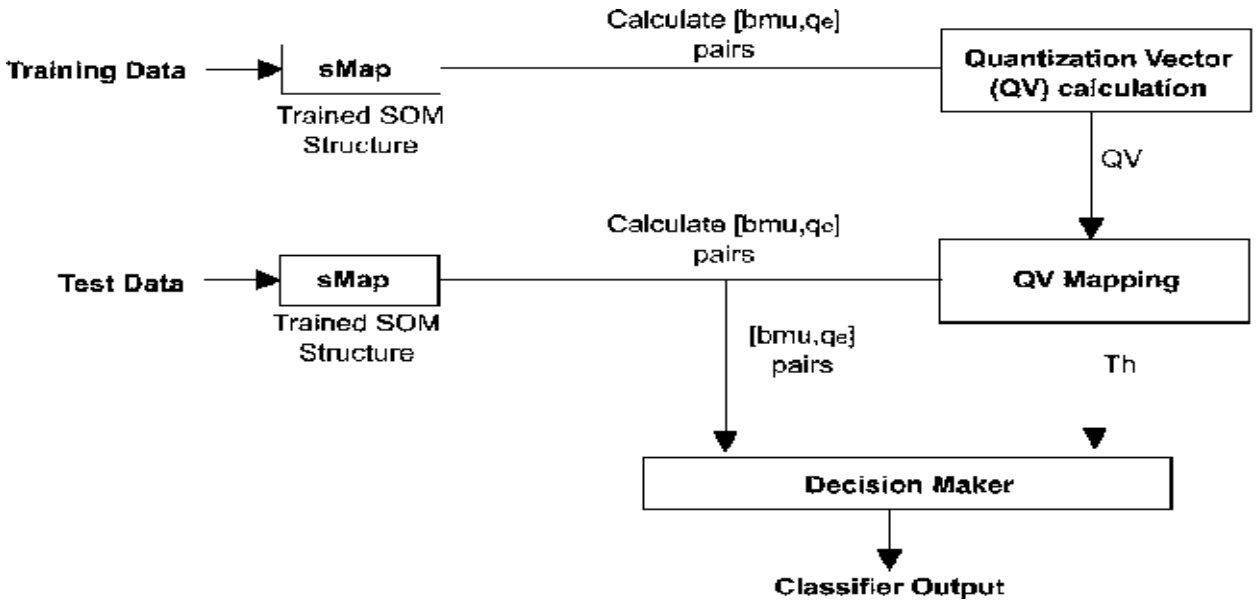
By comparing the latent representations of correlated stock pairs, autoencoders can be used to find them in the context of pairs trading. Stocks with similar latent representations are good candidates for pairs trading strategies because they are likely to show correlated behavior in their

price movements. Autoencoders help traders find potentially profitable trading opportunities by facilitating dimensionality reduction and anomaly detection by learning a compressed representation of the data. To model our scaled and encoded train and test datasets, we employed the layers and models functions found in Keras, a component of the Tensorflow libraries. We employed the 'adam' optimizer and the Relu activation function for both the input (encoded) and decoded layers. We utilized 'mse' as the loss function. To evaluate the model, we ran the Accuracy metric on both the train and test datasets.

3.9.5 Self Organising Maps (SOMs)

Kohonen maps, or self-organizing maps, are a kind of artificial neural network used for dimensionality reduction and unsupervised learning. Specifically designed to visualize and cluster high-dimensional data in a lower-dimensional space while maintaining the topological properties of the input data, Self Organizing Maps (SOMs) were created by Teuvo Kohonen in the 1980s. SOMs can be used in pairs trading to find correlated stock pairs by grouping stocks according to pertinent characteristics or past price movements. Traders can select pairs with high intra-cluster correlation for pairs trading strategies and identify groups of stocks that display similar price behaviors by examining the topology of the SOM grid.

SOMs use a two-dimensional grid of neurons to organize input data, with each neuron representing a cluster or prototype of the input data. Neighboring neurons are mapped to similar input vectors, reflecting the underlying topology of the input space in the arrangement of neurons on the grid. In order to train SOMs, the weights of the neurons are iteratively adjusted to match the distribution of the input data. The best matching unit (BMU) is determined by identifying the neuron whose weights are most similar to the input vectors when the SOM is being trained. Next, to get closer to the input vector in the input space, the weights of the BMU and the neurons that surround it are updated. SOMs' capacity to reduce high-dimensional input data's dimensionality while maintaining its essential qualities is one of its main features. SOMs offer a condensed representation of the input data that keeps its relationships and structure by arranging similar input vectors into adjacent neurons on the grid. SOMs can be visualized to learn more about the underlying structure of the input data after training. Similar weighted neurons have a tendency to group together on the grid, which makes it simple to spot input vector groups or clusters. Finding correlations, patterns, and outliers in the data can be aided by this visualization.



Input : *X={x1, x2, x3…xn}*

Step 1 : *Initialize weights for all nodes*

Step 2 : *A vector is chosen at random from the set of training data and presented into lattice*

Step 3 : *Calculate the distance between all input and output nodes using Euclidean Distance:*

$$d(i,j) = \sqrt{\sum_{j=1}^{N} (wij - xi)^2} \quad (1)$$

Step 4 : *The minimum d(i,j) is selected to be the winning unit*

Step 5 : *All weights for neighbor nodes are calculated using Gaussian Function*

Step 6 : *Each neighboring node's weight is adjusted to make it more likely the input vector using:*

$$w_{ij(t+1)} = w_{ij(t)} + \alpha \left(\left(\frac{\|rc - rk\|^2}{\delta} \right) - w_{ij(t)} \right) \quad (2)$$

Step 7 : *Update learning rate at certain time*

Step 8 : *Repeat Step 2 for N iterations*

Fig. 6. Self Organizing Maps Classifier framework (Anjum - 2022)

To model our train and test datasets, we imported the minisom function after installing the MiniSom Python library. We established a 10 by 10 SOM grid with a 0.5 learning rate, meaning

that there will be 10 rows and 10 columns in the SOM grid, or 100 neurons or units overall. The hyperparameter sigma=0.5 regulates the neighborhood function's width or spread in the SOM. The degree to which nearby neurons affect one another during training is determined by the neighborhood function. A narrower neighborhood, or one in which only nearby neurons significantly influence one another, is produced by a smaller value of sigma. On the other hand, a higher sigma value results in a wider neighborhood, which enables training to also impact neurons that are farther away from the winning neuron. The neighborhood function of the SOM will have a fairly moderate spread at sigma=0.5, allowing for the training of neighboring neurons to have a moderate influence. Sigma adjustments affect the SOM algorithm's convergence behavior and learning dynamics. The SOM was trained for 1,000 iterations. The best pairings were chosen using our correlation threshold of 0.8 after we had saved the best matching units (BMUs) in a data frame. We assessed the SOM model's performance in identifying these stock pairs using the Accuracy metric.

3.9.6 XGBoost

Extreme Gradient Boosting, or XGBoost for short, is an ensemble learning approach that combines boosting techniques with decision tree power (Chen Guestrin, 2016). Its efficacy, scalability, and capacity to handle intricate relationships within the data have led to a notable surge in its popularity. XGBoost excels at both classification and regression tasks and has been applied successfully in many domains, including finance.

System	exact greedy	approximate global	approximate local	out-of-core	sparsity aware	parallel
XGBoost	yes	yes	yes	yes	yes	yes
pGBRT	no	no	yes	no	no	yes
Spark MLLib	no	yes	no	no	partially	yes
H2O	no	yes	no	no	partially	yes
scikit-learn	yes	no	no	no	no	no
R GBM	yes	no	no	no	partially	no

Table 3. Comparison of major tree-boosting systems

Iterative optimization techniques are employed by the XGBoost algorithm to fine-tune the model's parameters and enhance its overall performance. We fed the preprocessed data into the XGBoost algorithm, and made necessary updates to the model's biases and weights, and then repeated the procedure until an acceptable outcome was obtained.

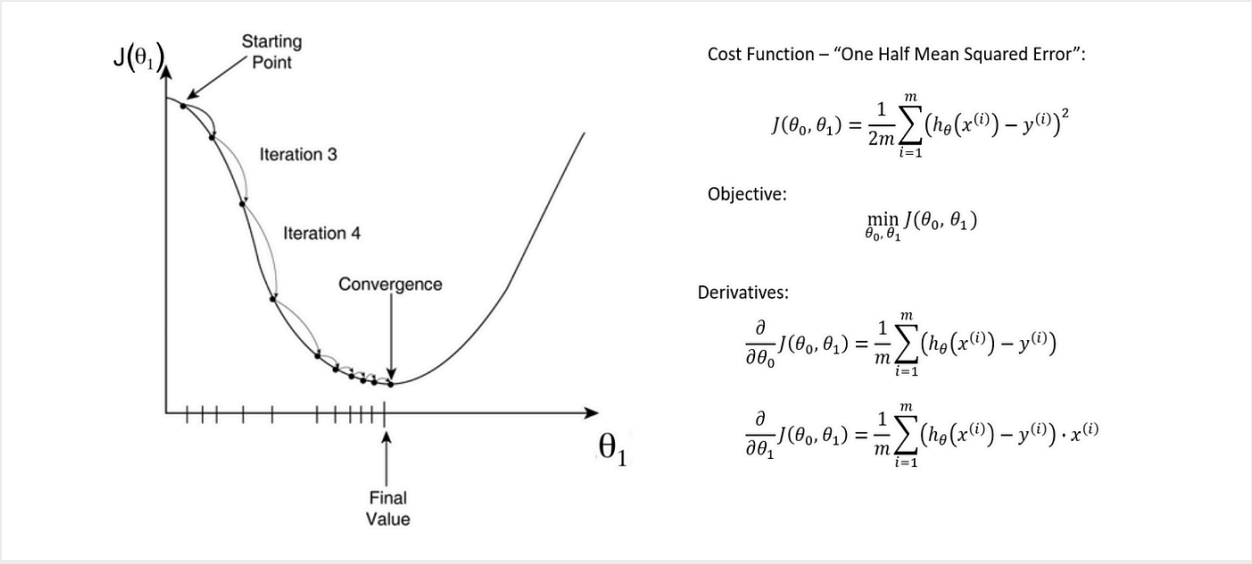


Fig. 7. Framework for Gradient Descent (KDnuggets - 2023)

3.10 Evaluation of classification algorithms

Since this is a classification effort, the effectiveness of the aforementioned models is assessed using a number of pertinent metrics that aid in measuring the effectiveness of classification algorithms. These consist of Accuracy, F1-score, Precision, and Recall.

The precision metric quantifies the percentage of actual positive cases among the positively predicted cases. It is particularly significant when false positives are expensive since it measures the classifier's accuracy in identifying positive instances, i.e. the cost of FP is much higher than FN, and the benefit of a TP is much higher than a TN.

Precision is calculated as: Precision = TP / (TP + FP)

where TP is the number of true positives, and FP is the number of false positives.

The percentage of true positive cases among actual positive instances is measured by recall, which is also referred to as sensitivity or true positive rate. It is particularly significant when false negatives are expensive since it measures the classifier's capacity to recognize every positive instance in the dataset.

Recall is calculated as: TP / (TP + FN)

where TP is the number of true positives, and FN is the number of false negatives.

The F1-score is a single metric that balances the classifier's ability to correctly identify positive instances and its ability to identify all the positive instances. It is calculated as the harmonic mean of precision and recall. When working with unbalanced datasets or when both false positives and false negatives are expensive, it is especially helpful.

F1-score is calculated as: F1-score = 2 * (Precision * Recall) / (Precision + Recall)

The ratio of accurate predictions—both true positives and true negatives—to the total number of occurrences is known as accuracy. It can be misleading due to imbalanced datasets.

Accuracy is calculated as: Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where TP = True Positives, TN = True Negatives, FP = False Positives and FN = False Negatives. Accuracy is used when the cost of FP and Fn are roughly equal, and the benefit of TP and TN are roughly equal.

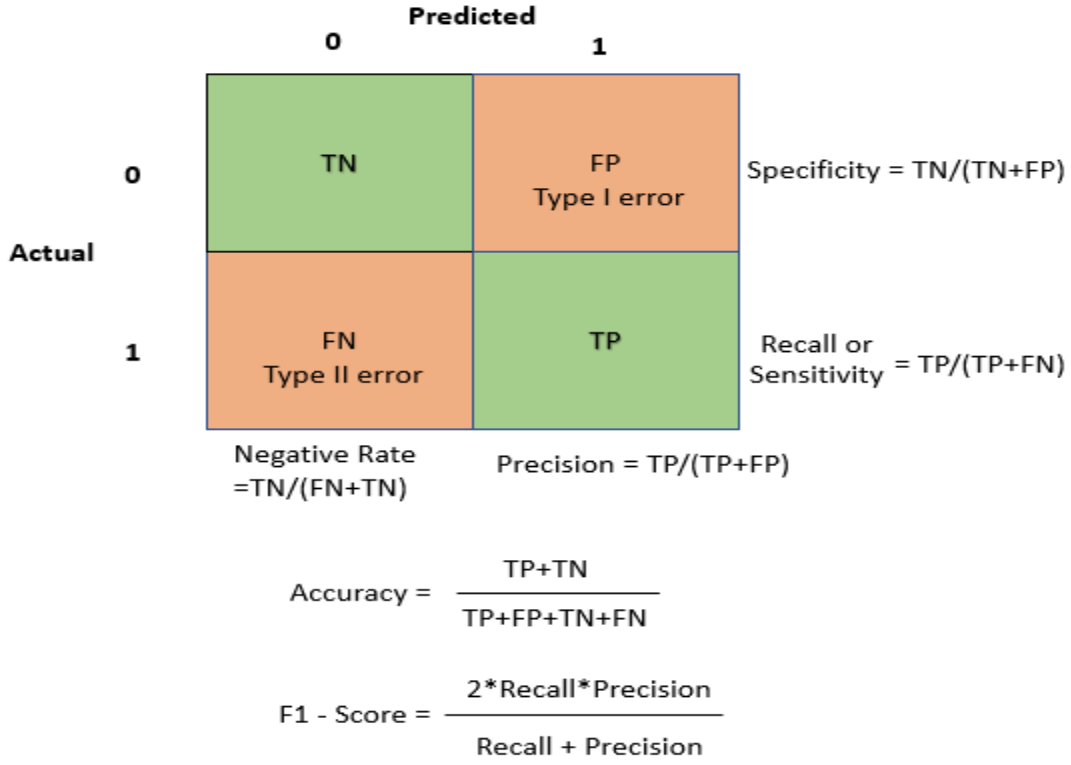


Fig. 8. Evaluation of Classification Algorithms

3.11 Implementing the Pairs Trading Strategy

We selected the best performing model and tuned its parameters via the GridSearchCV hyper parameter tuning method to enhance its performance. We evaluated the performance of the tuned

model, compared to the base model and where there was confirmed improvement, as judged by performance evaluation metrics of Accuracy, Recall and F1-Score, we used the model to execute the pairs trading strategy. We assumed an initial investment capital of USD 100,000. We calculated the spread between the 2 stock pairs. We then calculated the mean and standard deviation of the spread. The next step was to set the trade entry and exit thresholds. For our strategy, we selected an entry threshold of 1.0 and an exit threshold of 0.5. The entry threshold represents the level at which a trader initiates a position in a trading strategy. In this context, a value of 1.0 indicates that the strategy will enter a position when the distance between the correlated stocks falls below this threshold. A higher entry threshold implies a stricter criterion for entering a trade, potentially leading to fewer trading opportunities but with higher confidence in the trade's profitability. The exit threshold denotes the level at which a trader closes or exits a position in a trading strategy. Here, a value of 0.5 means that the strategy will exit a position when the distance between the correlated stocks rises above this threshold. A lower exit threshold implies a quicker exit from trades, potentially reducing profits from prolonged trends but also minimizing losses by capturing profits early or cutting losses swiftly. These thresholds are crucial parameters in defining the trading strategy's success and should be adjusted based on the trader's risk tolerance, market conditions, and specific trading objectives.

We then initiated variables for tracking trading positions and profits/ losses. The variables represent the current position or holding of the stocks in the pair in the trading strategy. A value of 0 indicates that no position is currently held for the specific stock. Depending on the trading strategy and market conditions, this variable will be updated to reflect long (positive value) or short (negative value) positions. We evaluated the performance of the pairs trading strategy by reviewing visual plots of the entry and exit points for each trading pair, and evaluated the profitability of the best performing (top 10) stock pairs.

3.12 Validating the Pairs Trading Strategy

Validating the tuned model requires us to provide previously unseen data and evaluate its performance on the new dataset. We chose the period January 1, 2023, to December 31, 2023, as our validation period in order to accomplish this. Using the stock pairs chosen by the model trained on the validation dataset, we evaluated the results of the pairs trading strategy by running the successful model on this dataset.

3.13 Model Deployment

We deployed our choice machine learning algorithm, XGBoost, within a pairs trading system using FastAPI, a modern Python web framework designed for creating APIs. The deployment process involved several key steps to enable users to access the model's predictions via HTTP requests. Firstly, the FastAPI instance was initialized to serve as the foundation for building API endpoints. The trained XGBoost model was then loaded into memory using the joblib.predict() function, enabling it to be readily accessible for making predictions. This pre-trained model, trained on historical stock data, serves as the predictive engine for identifying profitable trading pairs.

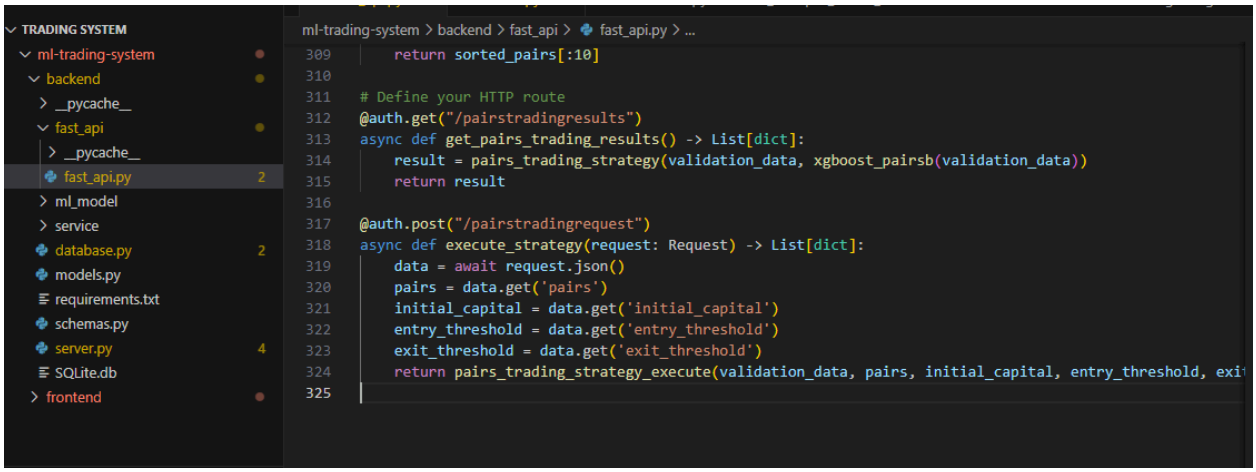


Fig.9 Fast API implementation

To structure the input and output of the API endpoints, six Pydantic models, namely pairstradingresults, pairs, users, login, register and pairstradingrequest are defined. The pairstradingrequest model specifies the required input parameters, which are the stock symbols, Initial Capital, Entry Threshold and Exit Threshold representing the trading pair. Conversely, the pairstradingresults model defines the structure of the response, including the predicted value for the trading pair. The users, login and register models handle the registration and authentication of a user into the system. The core functionality of the deployment revolves around the /pairstradingrequest/ endpoint, which is created using the @auth.post() decorator to denote that it accepts HTTP POST requests. When a POST request is received at this endpoint with the requisite input data (i.e., the stock symbols for the trading pair), the pairs_trading_strategy_execute() function is invoked. Before making predictions, input

validation is performed to ensure the validity of the provided stock symbols. If either symbol is invalid, an HTTP 400 Bad Request error is raised with an appropriate error message. Subsequently, the input data is prepared as an array, and the XGBoost model makes predictions based on this data. Finally, the predicted value is returned as part of the HTTP response in JSON format. This deployment setup enables users to interact with the trained XGBoost model, facilitating the integration of predictive capabilities into trading strategies within the pairs trading system.

Chapter 4: System Design and Architecture

This section describes the design and architecture of the system. The system automates stock pair selection using the developed XGBoost model and enables a trader to execute a pair trading strategy. Designing a pairs trading system involves several components, including data acquisition, strategy development, model training, execution, and user interface. The figure below depicts the relationship:

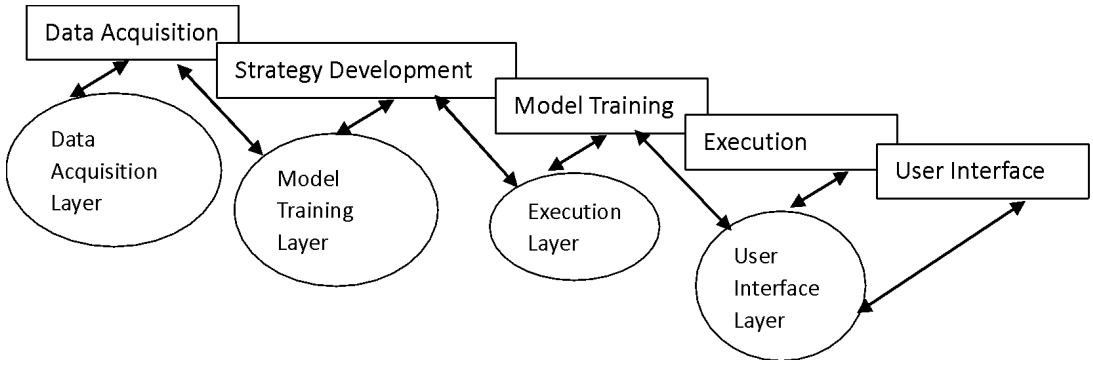


Fig. 10. System Design and Architecture

For data acquisition, we use the yfinance API wrapper to fetch closing price information for the 20 scoped stocks from the Yahoo Finance database. We developed a simple and intuitive web-based UI using React Javascript Framework that consumes the API for traders to interact with the system. The API loads the trained XGBoost model for stock pair selection, and enables traders to execute pairs trades over specified periods. It allows traders to input parameters (e.g. investment amount, risk tolerance) and customize trading settings. It displays relevant information, such as selected stock pairs, trading signals and performance metrics.

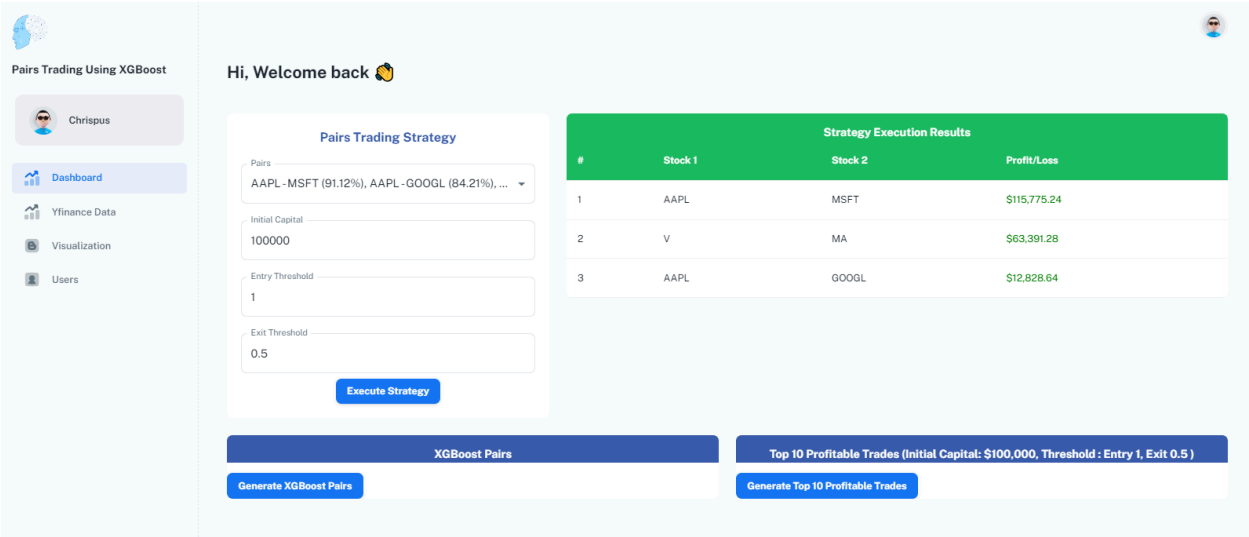


Fig. 11. Pairs Trading Strategy UI Dashboard

4.1 Model Deployment and Application Programming Interface (API) Development

Model deployment is a critical step in making machine learning models accessible for predictions in real-world applications. For the pairs trading system, deploying the XGBoost model enables traders to utilize it for predicting profitable stock pairs. Web service deployment using FastApi has been chosen for this project deployment, allowing the model to be exposed through an HTTP endpoint which is thereafter consumed by a React JS frontend. RESTful APIs provide a standardized and scalable way for clients to interact with the model using standard HTTP methods. This facilitates integration with other systems and applications, e.g the Yahoo Finance database, enabling seamless interaction with the model. Ensuring model persistence is crucial for deployment. The trained XGBoost model has been serialized and persisted in a format compatible with the deployment environment. The model has been saved using joblib.dump(). Security measures have also been implemented to protect the deployed model from unauthorized access and malicious attacks. This has been done using HTTPS for encrypted communication, authentication mechanisms i.e. JWT, and input validation to prevent injection attacks.

4.2 Deployment Strategy

Deploying the pairs trading system involved strategizing how to make the system available for use by traders while ensuring it is reliable, scalable, and maintainable. We deployed the model

using API. The exposed API endpoints were then consumed by the React JS framework. React framework is a javascript library used to build modern user interfaces based on reusable components. A number of considerations led us to decide to use an API to deploy our machine learning model, including the need for a flexible, modular, and easily maintained solution that would enable seamless integration with various client applications. The choice enabled us to separate the model's implementation from the maintenance of the Yahoo Finance database which houses updates to the stock prices, enabling both components to be updated and improved independently. Yahoo Finance serves the availability of the data via an API wrapper, called yfinance, enabling a seamless integration into the tool adopted to serve the output of this paper. Moreover, this approach facilitates scalability since the API can be designed to effectively handle several requests at once.

FastAPI's emphasis on speed, usability, and resilience makes it the ideal web framework for implementing our API. Developed on top of Pydantic for data handling and Starlette for the web parts, FastAPI provides asynchronous programming for better performance along with automated data validation, serialization, and documentation. FastAPI is a great option for deploying our machine-learning model in a scalable, maintainable, and production-ready manner because of these features.

4.3 API Design Principles

API design principles serve as the cornerstone for developing well-structured, intuitive, and maintainable APIs, essential for ensuring usability and comprehensibility. Maintaining consistency across all API endpoints is paramount, achieved through a uniform interface employing standard HTTP methods such as GET, POST, PUT, and DELETE (Fielding, 2000). Additionally, employing descriptive and meaningful naming conventions for endpoints, resources, and parameters enhances readability and usability, fostering consistency in API design (Roy, 2000). Striving for simplicity in API design is crucial, as it minimizes unnecessary complexity and provides only essential functionality, facilitating ease of integration for developers. Designing APIs with a clear and intuitive structure further minimizes the cognitive load on developers, ensuring ease of understanding and usage without extensive documentation. Designing APIs with performance optimization in mind minimizes latency and maximizes

throughput, utilizing efficient data formats and optimizing query parameters (Fielding, 2000). Implementing caching mechanisms to cache responses and reduce server load enhances scalability, controlled by HTTP caching headers to improve performance(Fielding, 2000; Roy, 2000; Garfinkel & Rosenberg, 2007).

The study followed several important design principles to ensure the security, scalability, and dependability of the API. It employs RESTful architecture first. Representational State Transfer (REST) is an architectural style that emphasizes scalability, statelessness, and a standardized interface for manipulating resources. Another key design principle is Microservices architecture. This is a modern approach to API design that decomposes applications into small, independent services. Each microservice focuses on a specific functionality or business domain, enabling teams to develop, deploy, and scale services independently. By embracing loose coupling and decentralized data management, microservices architecture fosters agility, resilience, and innovation (Newman, 2015). Lastly, the API makes use of asynchronous programming, which allows it to handle requests simultaneously without stalling, which enhances resource allocation and performance. Maintaining responsiveness requires asynchronous programming, particularly when there is a lot of traffic. By adhering to these API design principles, we sought to create an API that is user-friendly, reliable, and scalable, offering a positive experience for users.

4.4 API Architecture

API architecture encompasses the design principles and structural decisions that govern the development of an application programming interface (API). One common architectural approach is layered architecture, which divides the API into distinct layers: presentation, business logic, and data access. This separation of concerns promotes modularity and maintainability, allowing changes to one layer without affecting others (Roy, 2000). Another prevalent architectural style is RESTful architecture, which emphasizes resource-based interactions and statelessness. RESTful APIs represent resources as nouns and use HTTP methods to perform actions on these resources. By adhering to RESTful principles, APIs achieve scalability, reliability, and simplicity, enhancing their usability and interoperability (Fielding, 2000).

The data storage, endpoints, and server are the various parts that make up the API architecture. Incoming requests must be handled by the server, which must also forward them to the proper endpoints and reply. The foundation of server implementation is Starlette and Pydantic, upon which FastAPI is based. Endpoints serve as points of access for API functionality, providing particular resources for client interaction. Every endpoint bears the responsibility of handling the request and providing a response, in this case, the model predictions.

4.5 Framework and Tools

FastAPI stands out as a modern web framework ideal for constructing APIs with Python. It boasts high performance, asynchronous capabilities, and automatic API documentation generation, making it well-suited for real-time applications such as trading systems. FastAPI's integration with Pydantic ensures type safety and reduces boilerplate code, while its dependency injection system simplifies managing dependencies, fostering modular and reusable code (Sebastián Ramírez, 2022). In addition to FastAPI, the ReactJS library was used to build the user interface. It is based on javascript and typescript programming languages. Unlike the conventional ways used to build user interfaces by using languages such as HTML, the ReactJS library stands out due to its efficiency in reducing page loading times and real time rendering. The library will render and update the right components as the data changes. To make the development of this API easier, additional tools and libraries were used. These include the adaptable object Relational Mapper (ORM) for Python called SQLAlchemy. It is used to communicate more effectively and pythonically with the underlying relational SQLite database. It makes the process of modifying and querying data easier. The JSON Web Token (JWT) library was also used for handling authentication and authorization within the API, as a secure, scalable, and efficient system for managing user access.

XGBoost, a popular machine learning library, plays a pivotal role in the system by predicting the best stock pairs for trading based on historical data and features. Known for its efficient implementation and support for parallel processing, XGBoost is proficient in handling large datasets and training complex models (Chen & Guestrin, 2016).. With various hyperparameters available for fine-tuning model performance, including learning rate and regularization, the study demonstrated that XGBoost is a robust choice for implementing the pair selection tasks.

Finally, SQLite, a serverless and lightweight SQL database engine, was selected as the main database solution for storage and management. It is compatible with FastAPI, has good performance for small to medium-sized applications, and has the advantage of simplicity.

4.6 Model Integration

Model integration is a pivotal aspect of the pairs trading system, involving the seamless incorporation of the predictive model developed using XGBoost into the trading workflow. To ensure successful integration, several key steps were followed. Firstly, preprocessing of the data was essential to ensure compatibility with the model's input requirements. This entailed scaling, normalization, handling missing values, and feature engineering to create relevant input features for the model (Raschka & Mirjalili, 2019). Once the preprocessing was complete, the trained XGBoost model needed to be deployed within the trading system. This involved loading the serialized model using tools like JobLib and making it accessible within the application environment. The integrated model was capable of making predictions based on the provided data. The model predicts the best stock pairs for trading based on the historical data. The predictions generated by the model serve as inputs for making trading decisions within the system. Depending on the trading strategy implemented, the model's predictions may trigger buy, sell, or hold signals for specific stock pairs. Moreover, model integration should consider risk management strategies to mitigate potential losses (Chen & Guestrin, 2016).. For our system, this involved setting thresholds for position sizing, stop-loss orders, and portfolio diversification to manage risk exposure effectively. Before deploying the integrated model, thorough backtesting and validation are crucial. Backtesting involved simulating the model's performance on out-of-sample data to assess its effectiveness.

Continuous monitoring is essential post-deployment to ensure the model's performance remains consistent over time. Monitoring metrics such as accuracy, precision, recall, and profit/loss ratios will help identify any drift or degradation in model performance. Regular maintenance and updates may be necessary to retrain the model with new data periodically, incorporate feedback from trading experiences, and adapt to changing market conditions. The study integrated the trained model into the API by serializing the models to a suitable format by Python’s joblib

library, which converted the models into a binary format that could easily be stored and deployed.

4.7 API Endpoints

API endpoints serve as the gateway for clients to interact with the pairs trading system, offering access to specific functionalities and resources. Designing these endpoints requires careful consideration of several key factors. Firstly, endpoint design should adhere to RESTful principles, representing resources as nouns and utilizing HTTP methods like GET, POST, PUT, and DELETE for actions (Fielding, 2000). This ensures clarity and consistency in the API's structure and behavior. Intuitive and descriptive endpoint URLs enhance usability and understanding, while consistent naming conventions facilitate ease of use. Authentication and authorization mechanisms are crucial for ensuring secure access to sensitive resources and functionalities. Authentication verifies the identity of clients, while authorization determines their permissions based on roles and privileges. Common methods include API keys, OAuth 2.0, JWT tokens, and HTTP basic authentication. API endpoints typically support CRUD operations—Create, Read, Update, and Delete—for managing resources within the system. Create endpoints enable adding new data, Read endpoints retrieve data, Update endpoints modify existing data, and Delete endpoints remove data. These operations cover actions such as fetching data and executing trades.

Effective error handling is essential for providing clear and standardized responses when issues occur during request processing. Properly configured error responses, including appropriate HTTP status codes and descriptive messages, help clients understand and troubleshoot problems. Implementing error handling middleware ensures consistent error responses across all endpoints.
To manage server load and maintain system stability, rate limiting mechanisms should be implemented on API endpoints. These mechanisms restrict the number of requests clients can make within a specific time window, preventing abuse and ensuring fair usage. Rate limit headers in API responses inform clients of their remaining request quota and when they can make additional requests. Versioning support is crucial for maintaining backward compatibility and introducing changes without breaking existing clients. Endpoints should gracefully handle requests targeting deprecated versions and provide guidance for migrating to newer versions.

Versioning can be achieved through URL path prefixes (e.g., /v1/pairstradingrequest) or HTTP headers. Ensuring consistent behavior across API endpoints in various contexts and environments mitigates confusion and errors, maintaining predictability. Clear error messages and standardized error responses aid in diagnosing and troubleshooting issues effectively, further enhancing predictability. FastAPI endpoints are responsible for specific functionality, they accept input parameters in form of query parameters and return response parameters.

4.8 Authentication and Authorization

Robust authentication and authorization mechanisms safeguard sensitive data and resources (Garfinkel & Rosenberg, 2007). Ensuring data privacy and integrity through encryption protocols like HTTPS and encryption of data stored in databases protects against unauthorized access. Authentication and authorization mechanisms are vital components of the pairs trading system, ensuring secure access to sensitive resources and functionalities. Authentication serves as the initial step in verifying the identity of clients attempting to access the system. Common authentication methods include API keys, OAuth 2.0, JSON Web Tokens (JWT), and HTTP Basic Authentication. These methods authenticate clients by validating their credentials or tokens provided during the authentication process (Fielding, 2000). Following authentication, authorization determines the permissions granted to authenticated users based on their roles, privileges, or other attributes. Role-based access control (RBAC) and attribute-based access control (ABAC) are common authorization models employed in systems. RBAC assigns permissions to roles, while ABAC makes access decisions based on various attributes of the user, resource, and environment. Fine-grained access control enables administrators to define specific access rules for individual resources or actions, ensuring precise control over permissions (Raschka & Mirjalili, 2019).

Integration with identity providers (IdPs) e.g Gmail or Facebook, streamlines authentication and user management processes. Single sign-on (SSO) and federated authentication enhance user experience by allowing users to authenticate once and access multiple systems without repeated login prompts. Secure communication protocols such as HTTPS and Transport Layer Security (TLS) encrypt data transmitted between clients and the server, ensuring confidentiality, integrity, and authenticity of data exchanged over the network (Chen & Guestrin, 2016). Moreover,

auditing and logging mechanisms should be implemented to track authentication and authorization events. Audit logs provide a comprehensive trail of activities, including successful and failed login attempts, access to sensitive resources, and changes to access permissions. These logs play a crucial role in forensic analysis, compliance, and troubleshooting, aiding in the detection and mitigation of security incidents (Rebello & Carvalho, 2020). We used JWT and OAuth2PasswordBearer, a built-in FastAPI dependency, to implement authentication and authorization mechanisms in order to secure the API and guarantee that only authorized clients can access it. OAuth2PasswordBearer is an OAuth 2.0 extension that supports the grant type "Resource Owner Password Credentials," making it appropriate for first-party clients that can be trusted, like our web application. We were able to create a safe, scalable, and effective authentication and authorization system for our API by integrating JWT with OAuth2PasswordBearer.

Chapter 5: System Implementation and Testing

System Implementation and Testing represented a crucial phase in the development lifecycle of the pairs trading system, ensuring its functionality and adherence to specified requirements. During implementation, the system was developed according to the architectural design and requirements outlined in the system documentation. This involved writing code for various components such as API endpoints, data processing modules, prediction models, and user interface elements. Integration of these components was then carried out to create a cohesive system, linking API endpoints to backend services, integrating prediction models for stock pairing, and connecting user interface elements for seamless interaction. Additionally, database setup was performed to establish a structured storage environment for historical market data, user preferences, and trading strategies, ensuring efficient data retrieval and management.

In parallel with implementation, comprehensive testing processes were executed to validate the functionality, performance, and security of the system. Unit testing was conducted to verify the behavior of individual components in isolation, ensuring they operate as expected. Integration testing followed, validating interactions between different components to ensure seamless communication and functionality. End-to-end testing then assessed the entire system workflow, simulating user interactions, and verifying responses to ensure correctness. Performance testing evaluated the system's responsiveness and scalability under varying load conditions.

5.1 System Components

The pairs trading system comprises several integral components, each serving a specific role to ensure the system's functionality and efficiency. Firstly, the Data Acquisition component is responsible for sourcing market data from Yahoo Finance using the finance API wrapper. This data is then processed through Data Preprocessing modules, which clean, filter, and transform it into a structured format suitable for analysis and trading strategy development. The Prediction Model forms the other critical component, leveraging the machine learning algorithm, XGBoost, to identify the best trading pairs. Trading strategies are then devised utilizing the output of prediction model. Additionally, Risk Management metrics are integrated to manage risk exposure through strategies such as position sizing and stop-loss orders. User Interface components provide users with a dashboard to view stock data, trading signals, and performance

metrics. Configuration panels allow users to customize trading parameters, select strategies, and adjust risk management settings. Reporting and analytics tools enable the generation of reports, visualization of trading performance, and analysis of historical trading data.

The Database component serves as a repository for storing market data, user preferences, trading history, and system configurations. It also includes logging and audit trails to track system events, user activities, and ensure compliance with regulatory requirements. An Integration Layer facilitates interaction with external systems or users through API endpoints, enabling data retrieval and system configuration. Additionally, third-party integrations enhance system functionality by integrating with external services, APIs, or data providers.

5.2 Overview of the system architecture, depicting the interaction between these components

The system architecture of the pairs trading system orchestrates the interaction among its diverse components to facilitate efficient trading operations. At the outset, stockt data is fetched from the Yahoo Finance database, undergoes preprocessing to cleanse and structure it for analysis. Subsequently, the machine learning model, XGBoost, utilizes this data to predict the best stock pairs. These are then integrated into trading strategies, where the trading algorithm works in tandem with risk management modules to identify optimal trading decisions and manage risk exposure effectively. The figure below describes the system architecture, depicting the interaction between the components.

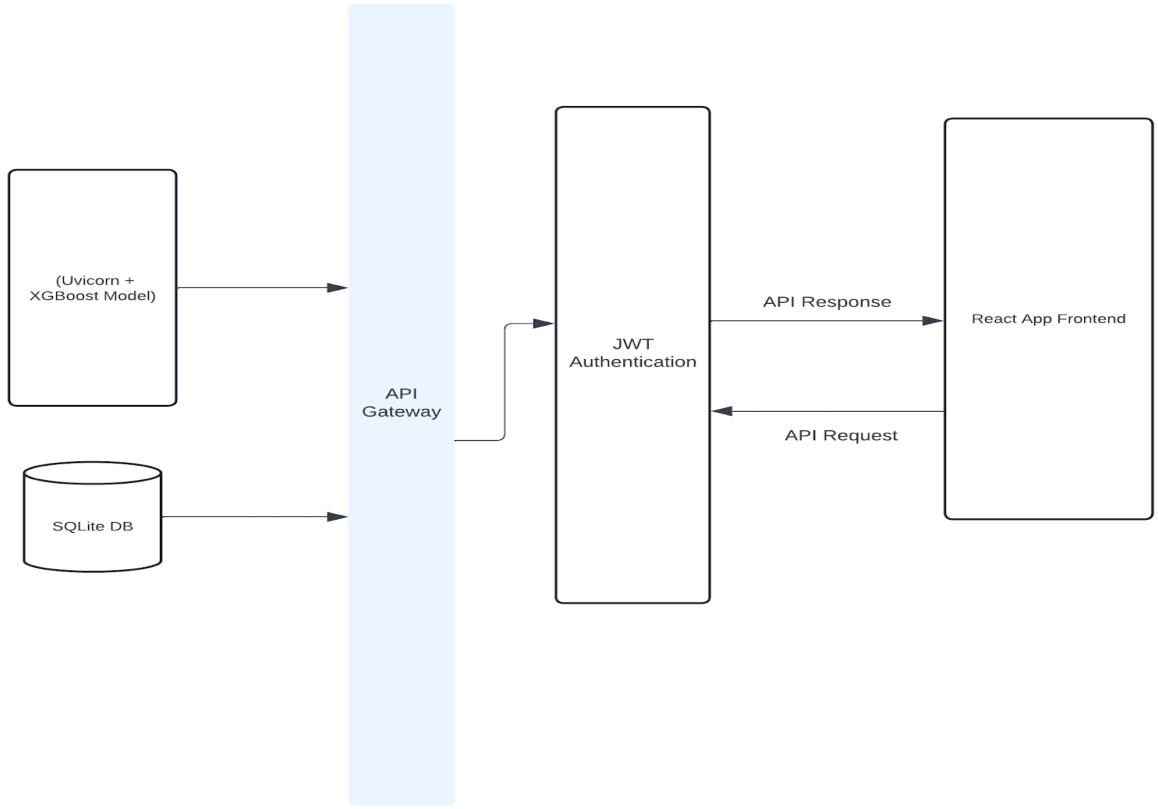


Fig. 12. Overview of the Pairs Trading System Architecture

Users interact with the system through a user interface comprising a dashboard for data visualization and a configuration panel for customization of trading parameters and risk management settings. Behind the scenes, data management is facilitated by a robust database storing both stock data and user-related information. Additionally, logging and audit trails capture system events and user activities for compliance and analysis purposes. The integration layer provides external access to the system via API endpoints, allowing for data retrieval, trade execution, and configuration management. Furthermore, third-party integrations enhance system functionality and provide access to stock data.

5.3 API User Registration

Machine learning services can be accessed securely thanks to the API. The application has multiple endpoints that enable different functionality and features in order to accomplish this. The registration process begins with users providing necessary information, including personal

details and authentication credentials. This input data undergoes validation to ensure accuracy, completeness, and compliance with registration requirements. API user registration involves the generation of unique API keys or tokens for each registered user to authenticate API requests.

User Registration:

In order to use the services, users must first create an account. To do this, the client sends a POST request to the registration endpoint with the user's details (password, email address, and username).

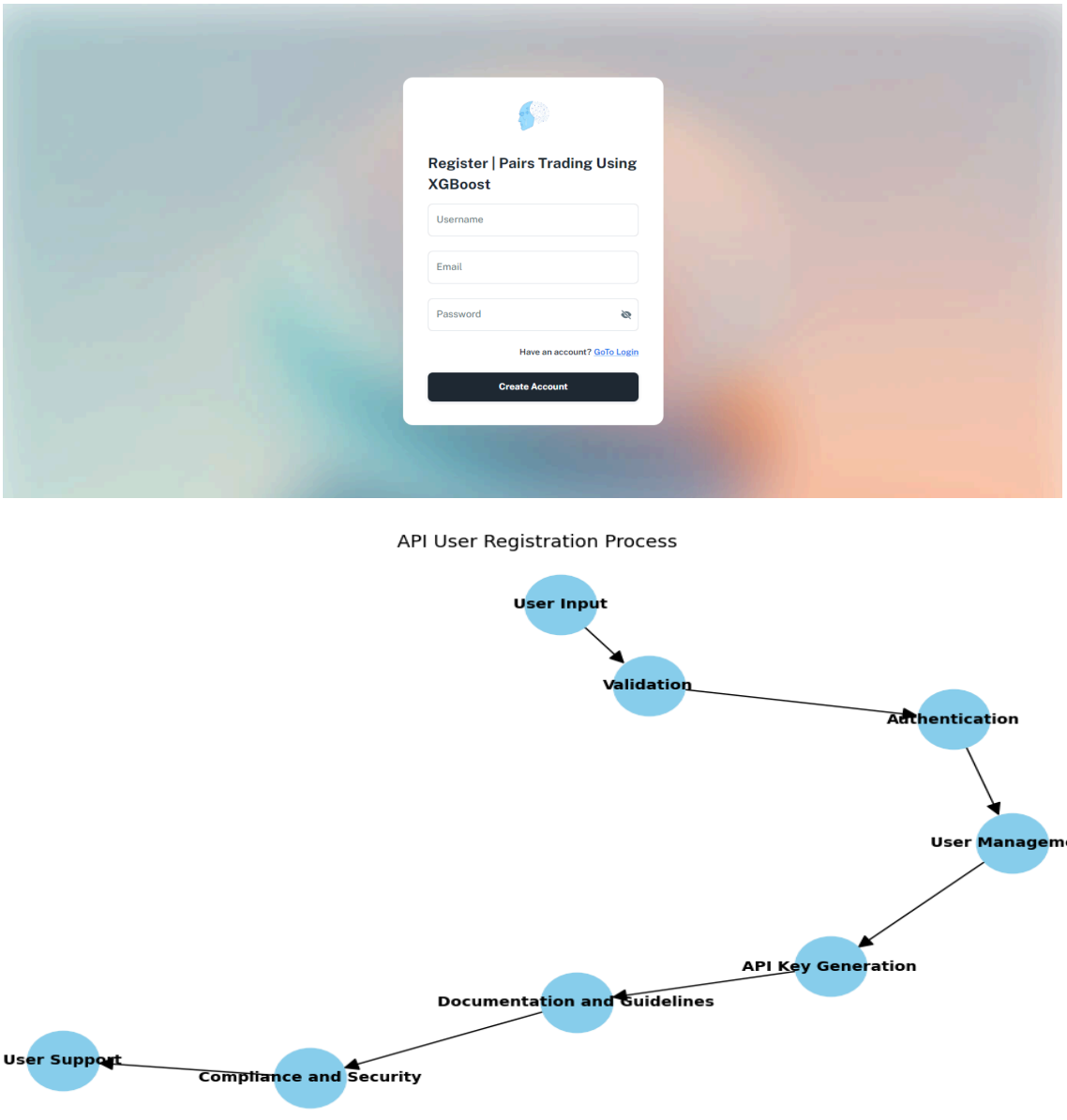


Fig. 13. User Registration Framework

The information is validated by the API server to make sure it follows the format specifications and that the password and username are distinct. Following a successful validation process, the user's password is hashed by the API server and saved in the SQLite database that houses and controls the user data. The information of the registered user is returned by the server, confirming that the registration was successful.

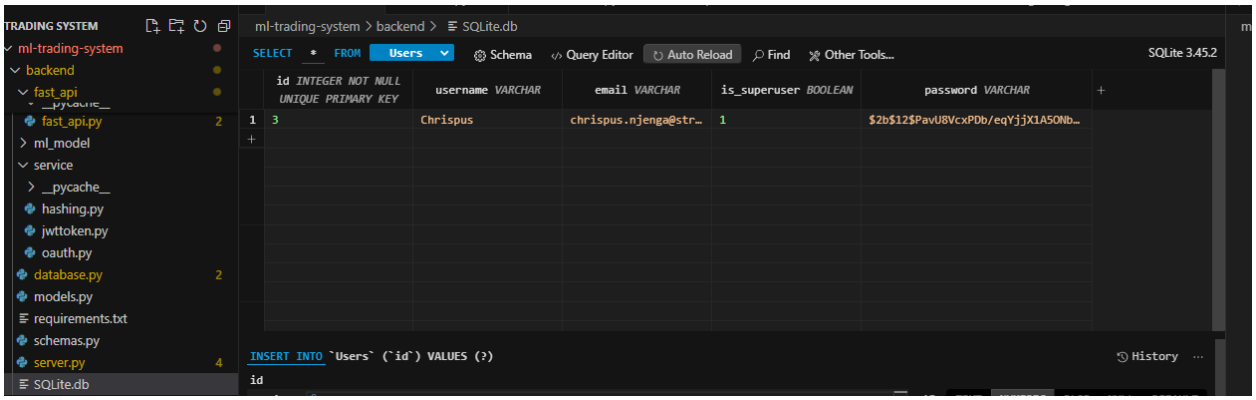


Fig. 14. User Details Database Table

5.4 API User Login and Token Generation

The user login endpoint serves as the gateway for users to authenticate themselves within the system. Access tokens play a critical role in authenticating subsequent API requests made by the user. A common approach for token generation is to use JSON Web Tokens (JWT). JWTs are digitally signed tokens comprising three parts: header, payload, and signature. The payload typically includes information such as the user's ID, username, and expiration time. After generating the JWT, the API signs it using a secret key known only to the server. The signed JWT is then returned to the client as the access token. This token serves as proof of authentication and is included in the Authorization header of subsequent requests to the API. The registered user can obtain a JWT for accessing protected endpoints and authenticate themselves using the Login User endpoints. The client accomplishes this by sending a POST request to the specified login endpoint with their credentials (password, email, and username). Once the user is successfully authenticated, the API generates a unique access token for the user's session. The endpoint verifies them against the stored user data in the system's database. If the provided credentials match those stored in the database, the user is considered authenticated.

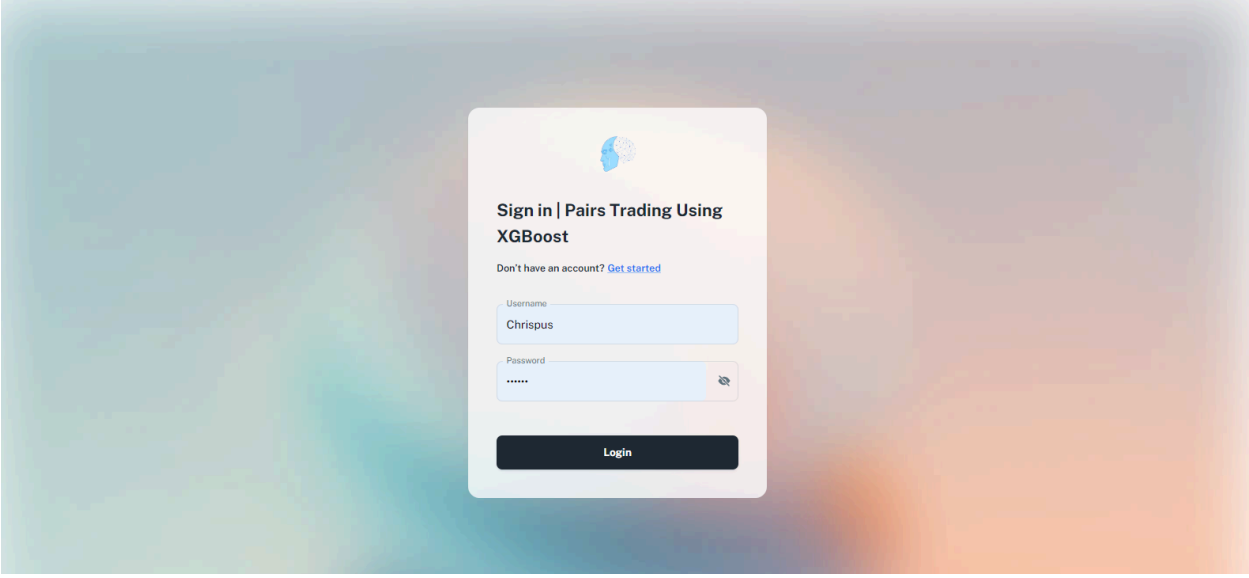


Fig. 15. User Sign In Page

5.5 Authentication and Authorization

JSON Web Tokens (JWT) and OAuth2 Password Bearer are used by the API server to handle authentication and authorization for the protected endpoints. A client requesting access to a protected endpoint needs to follow the Bearer scheme and include the previously obtained JWT in the Authorization header. After decoding this JWT, the API server verifies the validity of the signature and expiration timestamp. The user's ID and role are extracted by the server from the payload if the token is accepted. The server then contrasts the role of the user with the permissions needed to access the requested endpoint. The server keeps processing the request if the user has the necessary permissions.

5.6 Machine learning prediction endpoint

The machine learning prediction endpoint serves as a pivotal component within the system, facilitating the utilization of the trained machine learning model for making predictions. Its primary function revolves around processing incoming data and producing predictions through inference. Upon receiving a request, the prediction endpoint first preprocesses the input data to ensure compatibility with the model's input format. Subsequently, the preprocessed data is passed to the trained machine learning model, which generates predictions based on the provided input.

The result of this data is then consumed by the frontend react js library and displayed to the trader.

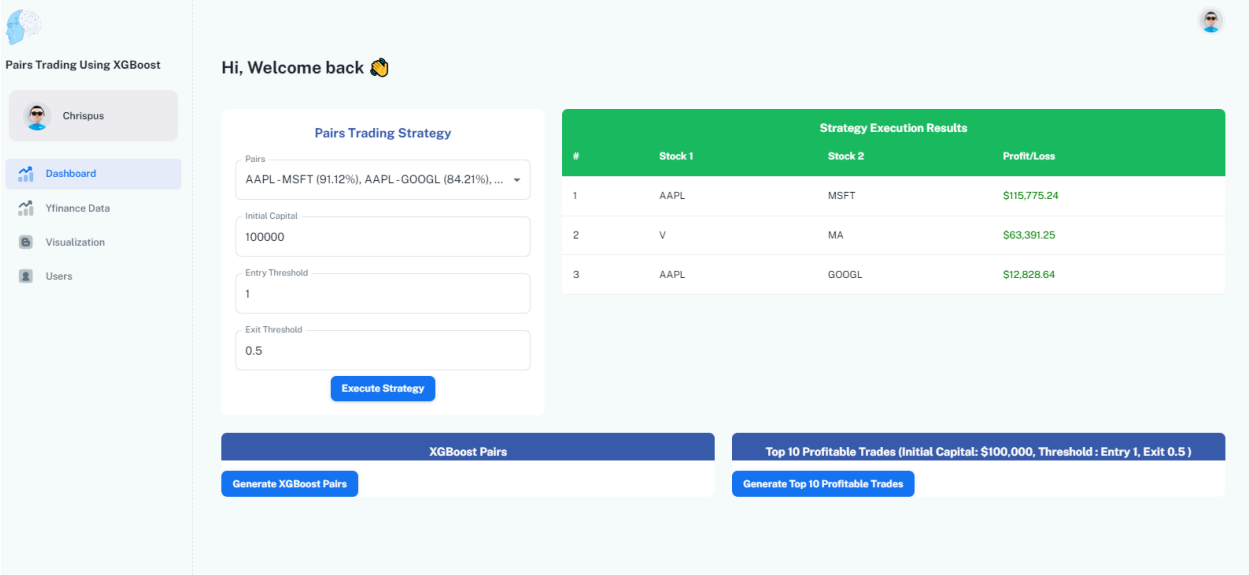


Fig. 16. Trading Strategy Output Page

5.7 SQLite Database

The selection and design of the database are crucial aspects of the system's architecture, as it serves as the foundation for storing and managing various types of data critical to the application's functionality. The choice of database type was influenced by factors such as data structure, scalability requirements, and the specific needs of the application. Common options include relational databases like MySQL or PostgreSQL, NoSQL databases such as MongoDB or Cassandra, and in-memory databases like Redis. These databases differ in their handling of data structure, scalability, and performance characteristics. Relational databases are well-suited for structured data with clearly defined relationships between entities, making them ideal for applications where data integrity and consistency are paramount. These are the key factors that led to the choice of the SQLite database for this project. SQLite Database conducts a number of functions and communicates with the API server. These responsibilities include keeping track of user data while they register, confirming their login credentials, and confirming their authorization to access protected endpoints. The hashed passwords, username, and user ID are stored in a table named users within the database schema.

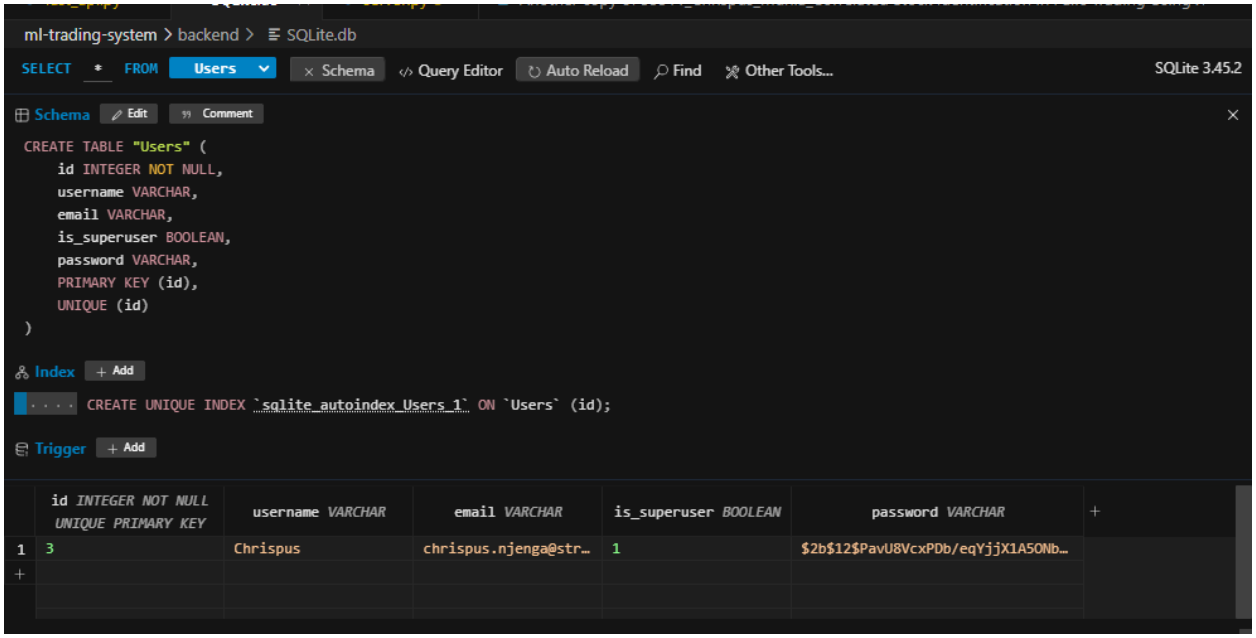


Fig. 17. View of the SQLite Database schema

Chapter 6: Results

The success of trading operations in pairs trading strategies is largely dependent on the identification of optimal pairs. Although there are a number of approaches available for finding compatible pairs, the goal of this research project was to use machine learning techniques to improve pair selection's precision and effectiveness. In particular, we sought to assess how well the XGBoost machine learning algorithm performed in identifying the optimal pairs for pairs trading strategies in comparison to other machine learning algorithms such as Random Forest, Support Vector Machines (SVM), Autoencoders, Self-Organizing Maps (SOMs), and others.

6.1 Exploratory Data Analysis (EDA)

6.1.1 Univariate Exploratory Analysis

We took the view that all relevant factors affecting the performance of a stock are reflected in the stock price, as represented by the Adjusted Closing Price. We observed evolution of the stock prices post recovery of the Great Financial Crisis of 2008 - 2009. We noted that the prices of the 20 stocks in the year 2010 were initially bunched up at levels below 100 dollars, most probably as a result of the stock market depression during the financial crisis, but diverged for some stocks especially from 2015 in line with sustained strong economic recovery in the US.

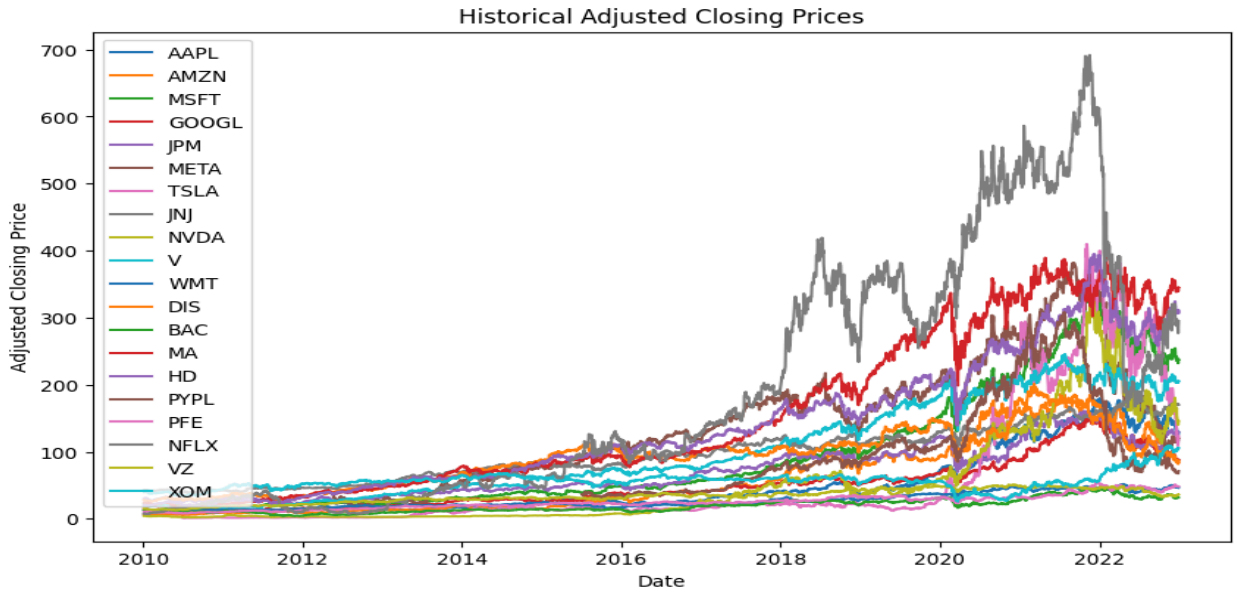


Fig. 18. Historical Adjusted closing prices for the 20 stocks used in the modeling

6.1.2 Multivariate Exploratory Analysis

Using Euclidean distance method, we sought to understand the multivariate relationship between the 20 stocks included in the research. It was clear from the plot that the set of stocks was appropriate for the analysis as they did not have distances that varied greatly (greater than 50).

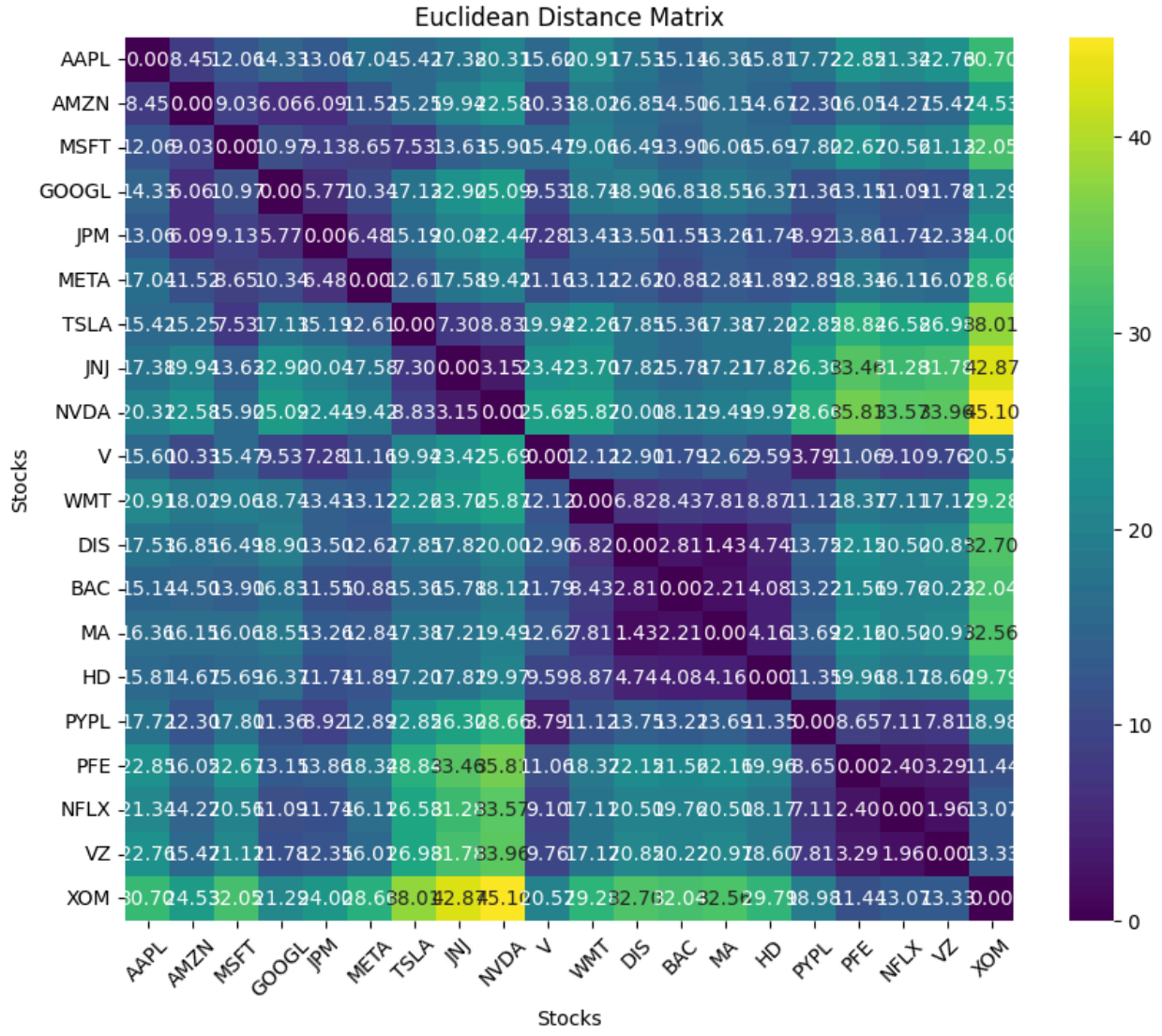


Fig. 19. Multivariate Analysis of the stocks universe

6.2 Model Performance Evaluation

6.2.1 Distance Approach model

The model sought to calculate the distance between each pair of securities based on their historical price movements. We set a threshold value of 0.8 to identify pairs with sufficiently

high similarity or correlation based on the calculated distances. Based on this threshold, the model did not select any tradable pairs. We thus dropped the model from the analysis.

6.2.2 Random Forest Classifier

The Random Forest Classifier is an ensemble learning technique that builds a set of decision trees and uses majority voting to combine the predictions made by the trees in order to reduce overfitting and enhance overall performance. It offers a natural measure of feature importance, is resilient to noise, and can handle big datasets. On the other hand, compared to single decision trees, Random Forest Classifier models might be more computationally expensive and less interpretable. On running the train and test datasets on the RandomForestClassifier model, we obtained the below results, using the classification_report function imported from the sklearn.metrics Python library:

	Precision	**Recall**	**F1 - Score**	**Support**
0	0.97	0.94	0.95	198
1	0.94	0.97	0.95	180
Accuracy			0.95	378
Macro Avg	0.95	0.95	0.95	378
Weighted Avg	0.95	0.95	0.95	378

Table 4. Random Forest Classifier Classification Report

We then used the model to obtain tradable pairs stocks and obtained below pairs.

```
Random Forest selected pairs: [('AAPL', 'MSFT'), ('AAPL', 'NVDA'),
('AAPL', 'HD'), ('AAPL', 'PFE'), ('MSFT', 'GOOGL'), ('MSFT', 'NVDA'),
('MSFT', 'BAC'), ('MSFT', 'HD'), ('MSFT', 'PFE'), ('GOOGL', 'META'),
('GOOGL', 'NVDA'), ('GOOGL', 'BAC'), ('GOOGL', 'HD'), ('GOOGL', 'PFE'),
('JPM', 'BAC'), ('JPM', 'XOM'), ('TSLA', 'NFLX'), ('NVDA', 'HD'), ('NVDA',
'PFE'), ('BAC', 'HD'), ('BAC', 'XOM'), ('HD', 'PFE'), ('AMZN', 'V'),
('AMZN', 'WMT'), ('AMZN', 'BAC'), ('AMZN', 'MA'), ('AMZN', 'HD'), ('AMZN',
'PYPL'), ('AMZN', 'NFLX'), ('AMZN', 'VZ'), ('MSFT', 'GOOGL'), ('MSFT',
'JPM'), ('MSFT', 'TSLA'), ('MSFT', 'JNJ'), ('MSFT', 'NVDA'), ('MSFT',
'V'), ('MSFT', 'WMT'), ('MSFT', 'BAC'), ('MSFT', 'MA'), ('MSFT', 'HD'),
('MSFT', 'PFE'), ('GOOGL', 'JPM'), ('GOOGL', 'TSLA'), ('GOOGL', 'JNJ'),
('GOOGL', 'NVDA'), ('GOOGL', 'V'), ('GOOGL', 'WMT'), ('GOOGL', 'BAC'),
```

```
('GOOGL', 'MA'), ('GOOGL', 'HD'), ('GOOGL', 'PFE'), ('JPM', 'META'),
('JPM', 'JNJ'), ('JPM', 'NVDA'), ('JPM', 'V'), ('JPM', 'WMT'), ('JPM',
'BAC'), ('JPM', 'MA'), ('JPM', 'HD'), ('JPM', 'NFLX'), ('META', 'DIS'),
('META', 'PYPL'), ('META', 'NFLX'), ('TSLA', 'JNJ'), ('TSLA', 'NVDA'),
('TSLA', 'WMT'), ('TSLA', 'HD'), ('TSLA', 'PFE'), ('JNJ', 'NVDA'), ('JNJ',
'V'), ('JNJ', 'WMT'), ('JNJ', 'BAC'), ('JNJ', 'MA'), ('JNJ', 'HD'),
('JNJ', 'PFE'), ('NVDA', 'V'), ('NVDA', 'WMT'), ('NVDA', 'BAC'), ('NVDA',
'MA'), ('NVDA', 'HD'), ('NVDA', 'PFE'), ('V', 'WMT'), ('V', 'BAC'), ('V',
'MA'), ('V', 'HD'), ('V', 'PYPL'), ('V', 'NFLX'), ('V', 'VZ'), ('WMT',
'BAC'), ('WMT', 'MA'), ('WMT', 'HD'), ('WMT', 'PFE'), ('DIS', 'PYPL'),
('BAC', 'MA'), ('BAC', 'HD'), ('BAC', 'PFE'), ('MA', 'HD'), ('MA',
'NFLX'), ('MA', 'VZ'), ('HD', 'PFE'), ('PYPL', 'NFLX')]
```

Below Feature Importance graph shows the stocks having the greatest influence in the Random Forest Classifier model.

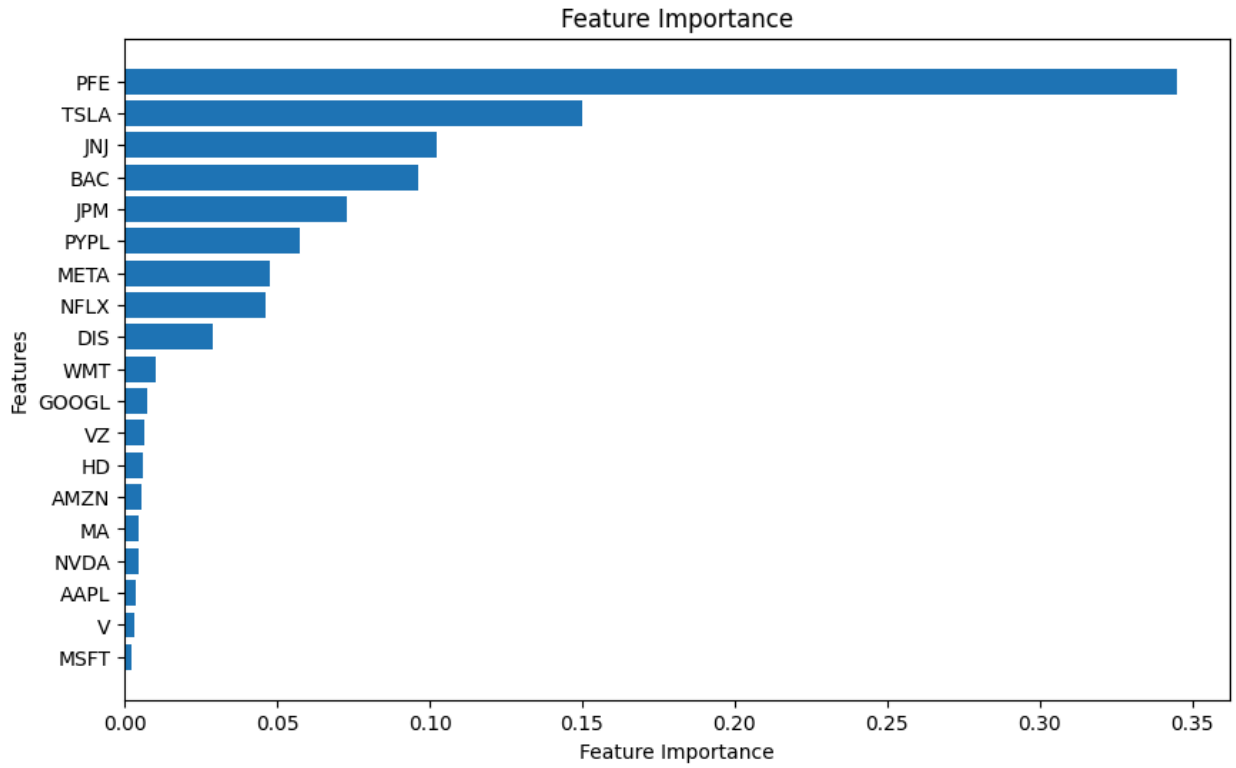


Fig. 20. Feature Importance for the Random Forest Classifier Model

6.2.3 Support Vector Machine (SVM) Classifier

The Support Vector Machine classifier is a tree-based algorithm that divides the feature space recursively to produce a hierarchical structure. A class label is associated with each leaf node, and each node represents a decision rule based on a single feature. They can handle both

continuous and categorical features, and they are interpretable and noise-resistant. On running the train and test datasets on the SVM Classifier model, we obtained the below results, using the classification_report function imported from the sklearn.metrics Python library:

	Precision	**Recall**	**F1 - Score**	**Support**
0	0.96	0.75	0.84	198
1	1.78	0.97	0.86	180
Accuracy			0.85	378
Macro Avg	0.87	0.86	0.85	378
Weighted Avg	0.88	0.85	0.85	378

Table 5. SVM Classifier Classification Report

We then used the model to obtain tradable pairs stocks and obtained below pairs.

```
SVM selected pairs: [('MSFT', 'NVDA'), ('MSFT', 'V'), ('MSFT', 'WMT'),
('MSFT', 'BAC'), ('MSFT', 'MA'), ('MSFT', 'HD'), ('MSFT', 'PFE'),
('GOOGL', 'JPM'), ('GOOGL', 'TSLA'), ('GOOGL', 'JNJ'), ('GOOGL', 'NVDA'),
('GOOGL', 'V'), ('GOOGL', 'WMT'), ('GOOGL', 'BAC'), ('GOOGL', 'MA'),
('GOOGL', 'HD'), ('GOOGL', 'PFE'), ('JPM', 'META'), ('JPM', 'JNJ'),
('JPM', 'NVDA'), ('JPM', 'V')('AAPL', 'AMZN'), ('AAPL', 'MSFT'), ('AAPL',
'GOOGL'), ('AAPL', 'JPM'), ('AAPL', 'TSLA'), ('AAPL', 'JNJ'), ('AAPL',
'NVDA'), ('AAPL', 'V'), ('AAPL', 'WMT'), ('AAPL', 'BAC'), ('AAPL', 'MA'),
('AAPL', 'HD'), ('AAPL', 'PFE'), ('AMZN', 'MSFT'), ('AMZN', 'GOOGL'),
('AMZN', 'JPM'), ('AMZN', 'META'), ('AMZN', 'TSLA'), ('AMZN', 'JNJ'),
('AMZN', 'NVDA'), ('AMZN', 'V'), ('AMZN', 'WMT'), ('AMZN', 'BAC'),
('AMZN', 'MA'), ('AMZN', 'HD'), ('AMZN', 'PYPL'), ('AMZN', 'NFLX'),
('AMZN', 'VZ'), ('MSFT', 'GOOGL'), ('MSFT', 'JPM'), ('MSFT', 'TSLA'),
('MSFT', 'JNJ'), ('JPM', 'WMT'), ('JPM', 'BAC'), ('JPM', 'MA'), ('JPM',
'HD'), ('JPM', 'NFLX'), ('META', 'DIS'), ('META', 'PYPL'), ('META',
'NFLX'), ('TSLA', 'JNJ'), ('TSLA', 'NVDA'), ('TSLA', 'WMT'), ('TSLA',
'HD'), ('TSLA', 'PFE'), ('JNJ', 'NVDA'), ('JNJ', 'V'), ('JNJ', 'WMT'),
('JNJ', 'BAC'), ('JNJ', 'MA'), ('JNJ', 'HD'), ('JNJ', 'PFE'), ('NVDA',
'V'), ('NVDA', 'WMT'), ('NVDA', 'BAC'), ('NVDA', 'MA'), ('NVDA', 'HD'),
('NVDA', 'PFE'), ('V', 'WMT'), ('V', 'BAC'), ('V', 'MA'), ('V', 'HD'),
('V', 'PYPL'), ('V', 'NFLX'), ('V', 'VZ'), ('WMT', 'BAC'), ('WMT', 'MA'),
('WMT', 'HD'), ('WMT', 'PFE'), ('DIS', 'PYPL'), ('BAC', 'MA'), ('BAC',
'HD'), ('BAC', 'PFE'), ('MA', 'HD'), ('MA', 'NFLX'), ('MA', 'VZ'), ('HD',
'PFE'), ('PYPL', 'NFLX')]
```

Below Feature Importance graph shows the stocks having the greatest influence in the SVM Classifier model.

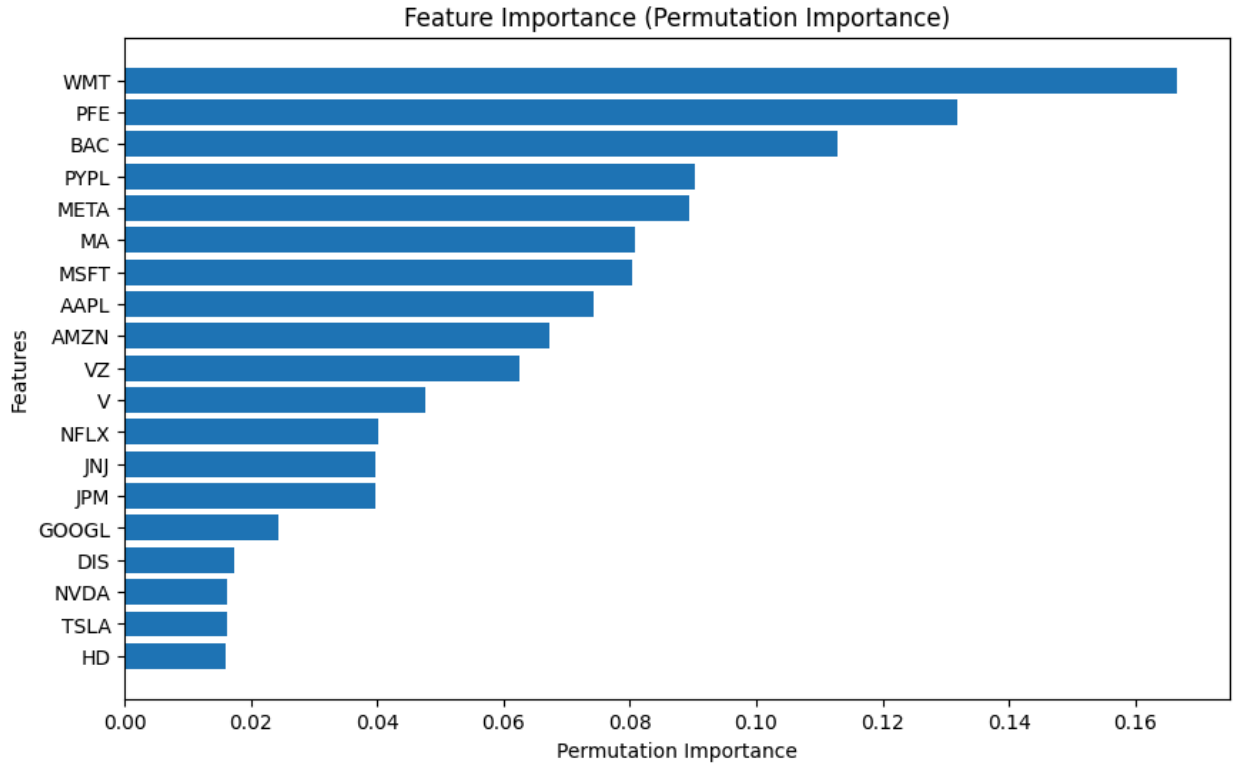


Fig. 21. Feature Importance for the SVM Classifier Model

6.2.4 Autoencoder Classifier Model

We ran the train and test data through our autoencoder model. We used the layers and models functions in Keras (part of the Tensorflow libraries) to model our scaled and encoded train and test datasets. Setting the threshold at 1, the model did not return any tradable stock pairs. The threshold is used to determine the maximum acceptable distance between pairs of stocks. If the distance between two stocks (represented by their encoded representations) is less than the threshold, it implies that these stocks are considered close or similar according to the autoencoder's learned representations. If the threshold is set to a low value (e.g., 0.5), only pairs of stocks with very similar encoded representations will be selected. If the threshold is set to a high value (e.g., 2.0), more pairs of stocks, even those with slightly different encoded representations, will be selected. Therefore, the threshold serves as a parameter to control the sensitivity of the selection process. A lower threshold results in more stringent selection criteria, whereas a higher threshold allows for more diverse pairs of stocks to be selected. Adjusting the

threshold allows traders to customize the selection process based on their preferences and risk tolerance. Evaluating the performance of the Autoencoder model, we obtained a Train Accuracy of 0.878145 and a Test Accuracy of 0.862434.

6.2.5 Self Organizing Maps

We trained the SOM model using the training dataset and visualized the output to gain insights into the underlying pairs selection. Neurons with similar weights tend to cluster together on the grid, making it easy to identify groups or clusters of similar input vectors. This visualization can help identify patterns, outliers, and correlations within the data.

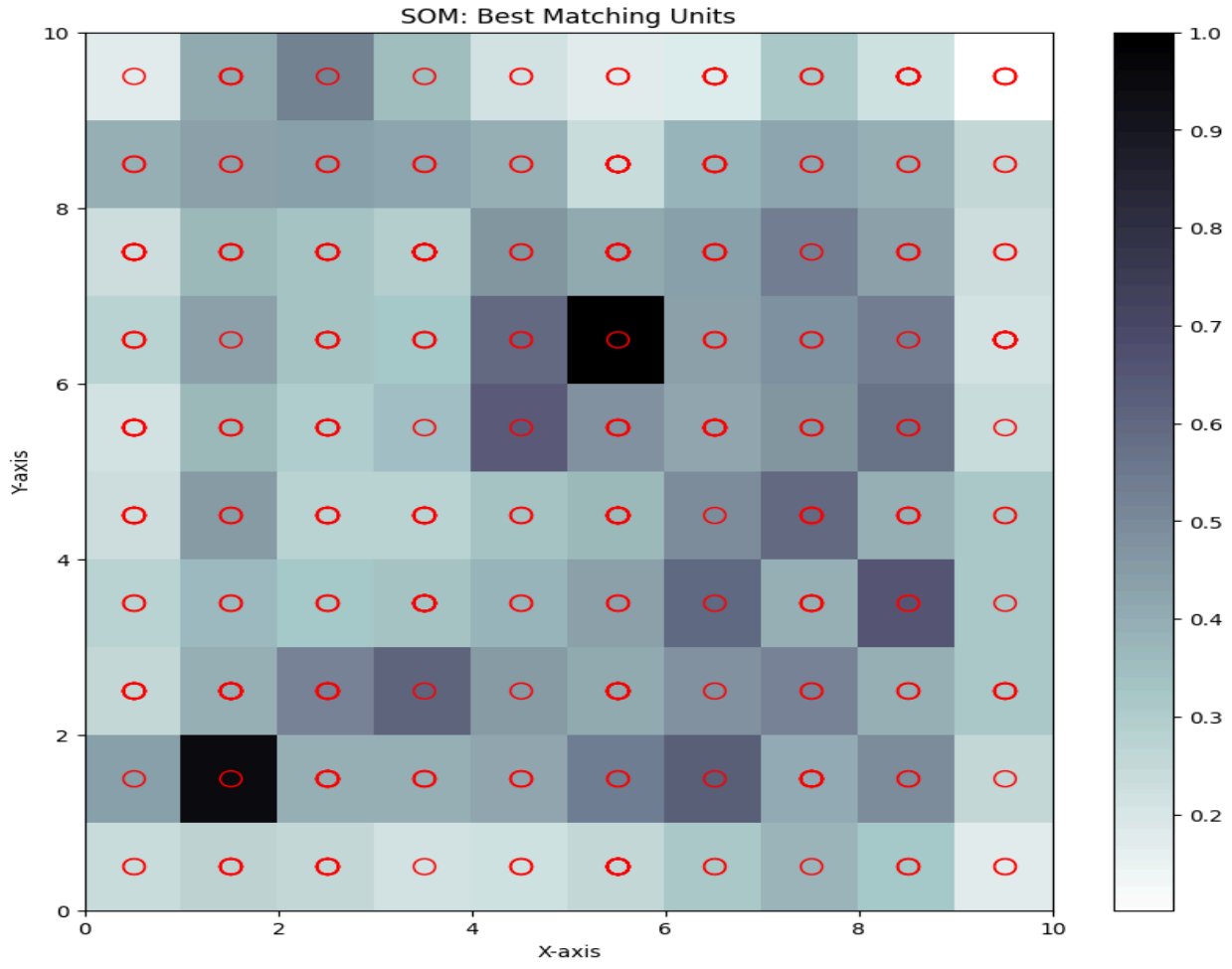


Fig. 22. SOM Best Matching Units (BMUs)

We then evaluated the performance of the SOM model and obtained an Accuracy result of `0.54497354`. Based on this low performance metric, we did not proceed to select tradable pairs and disqualified the model as an effective algorithm for stock pair selection.

6.2.6 XGBoost

XGBoost is a combination of regularization and gradient boosting techniques that provides robustness against overfitting and high predictive accuracy (Chen and Guestrin, 2016). The algorithm uses an improved tree-boosting framework (XGBoost (eXtreme Gradient Boosting), 2023) that combines parallel computing and tree pruning to achieve remarkable results. On running the train and test datasets on the XGBClassifier model, we obtained the below results, using the classification_report function imported from the sklearn.metrics Python library:

	Precision	**Recall**	**F1 - Score**	**Support**
0	0.97	0.94	0.96	198
1	0.94	0.97	0.95	180
Accuracy			0.96	378
Macro Avg	0.95	0.96	0.95	378
Weighted Avg	0.96	0.96	0.96	378

Table 6. XG Boost Classifier Classification Report

Given the strong comparative model performance, we used the model to obtain tradable pairs stocks and obtained below pairs.

```
XGBoost selected pairs: [('AAPL', 'AMZN'), ('AAPL', 'MSFT'), ('AAPL',
'GOOGL'), ('AAPL', 'JPM'), ('AAPL', 'TSLA'), ('AAPL', 'JNJ'), ('AAPL',
'NVDA'), ('AAPL', 'V'), ('AAPL', 'WMT'), ('AAPL', 'BAC'), ('AAPL', 'MA'),
('AAPL', 'HD'), ('AAPL', 'PFE'), ('AMZN', 'MSFT'), ('AMZN', 'GOOGL'),
('AMZN', 'JPM'), ('AMZN', 'META'), ('AMZN', 'TSLA'), ('AMZN', 'JNJ'),
('AMZN', 'NVDA'), ('AMZN', 'V'), ('AMZN', 'WMT'), ('AMZN', 'BAC'),
('AMZN', 'MA'), ('AMZN', 'HD'), ('AMZN', 'PYPL'), ('AMZN', 'NFLX'),
('AMZN', 'VZ'), ('MSFT', 'GOOGL'), ('MSFT', 'JPM'), ('MSFT', 'TSLA'),
('MSFT', 'JNJ'), ('MSFT', 'NVDA'), ('MSFT', 'V'), ('MSFT', 'WMT'),
('MSFT', 'BAC'), ('MSFT', 'MA'), ('MSFT', 'HD'), ('MSFT', 'PFE'),
('GOOGL', 'JPM'), ('GOOGL', 'TSLA'), ('GOOGL', 'JNJ'), ('GOOGL', 'NVDA'),
('GOOGL', 'V'), ('GOOGL', 'WMT'), ('GOOGL', 'BAC'), ('GOOGL', 'MA'),
('GOOGL', 'HD'), ('GOOGL', 'PFE'), ('JPM', 'META'), ('JPM', 'JNJ'),
```

```
('JPM', 'NVDA'), ('JPM', 'V'), ('JPM', 'WMT'), ('JPM', 'BAC'), ('JPM',
'MA'), ('JPM', 'HD'), ('JPM', 'NFLX'), ('META', 'DIS'), ('META', 'PYPL'),
('META', 'NFLX'), ('TSLA', 'JNJ'), ('TSLA', 'NVDA'), ('TSLA', 'WMT'),
('TSLA', 'HD'), ('TSLA', 'PFE'), ('JNJ', 'NVDA'), ('JNJ', 'V'), ('JNJ',
'WMT'), ('JNJ', 'BAC'), ('JNJ', 'MA'), ('JNJ', 'HD'), ('JNJ', 'PFE'),
('NVDA', 'V'), ('NVDA', 'WMT'), ('NVDA', 'BAC'), ('NVDA', 'MA'), ('NVDA',
'HD'), ('NVDA', 'PFE'), ('V', 'WMT'), ('V', 'BAC'), ('V', 'MA'), ('V',
'HD'), ('V', 'PYPL'), ('V', 'NFLX'), ('V', 'VZ'), ('WMT', 'BAC'), ('WMT',
'MA'), ('WMT', 'HD'), ('WMT', 'PFE'), ('DIS', 'PYPL'), ('BAC', 'MA'),
('BAC', 'HD'), ('BAC', 'PFE'), ('MA', 'HD'), ('MA', 'NFLX'), ('MA', 'VZ'),
('HD', 'PFE'), ('PYPL', 'NFLX')]
```

Below Feature Importance graph shows the stocks having the greatest influence in the XGB Classifier model.

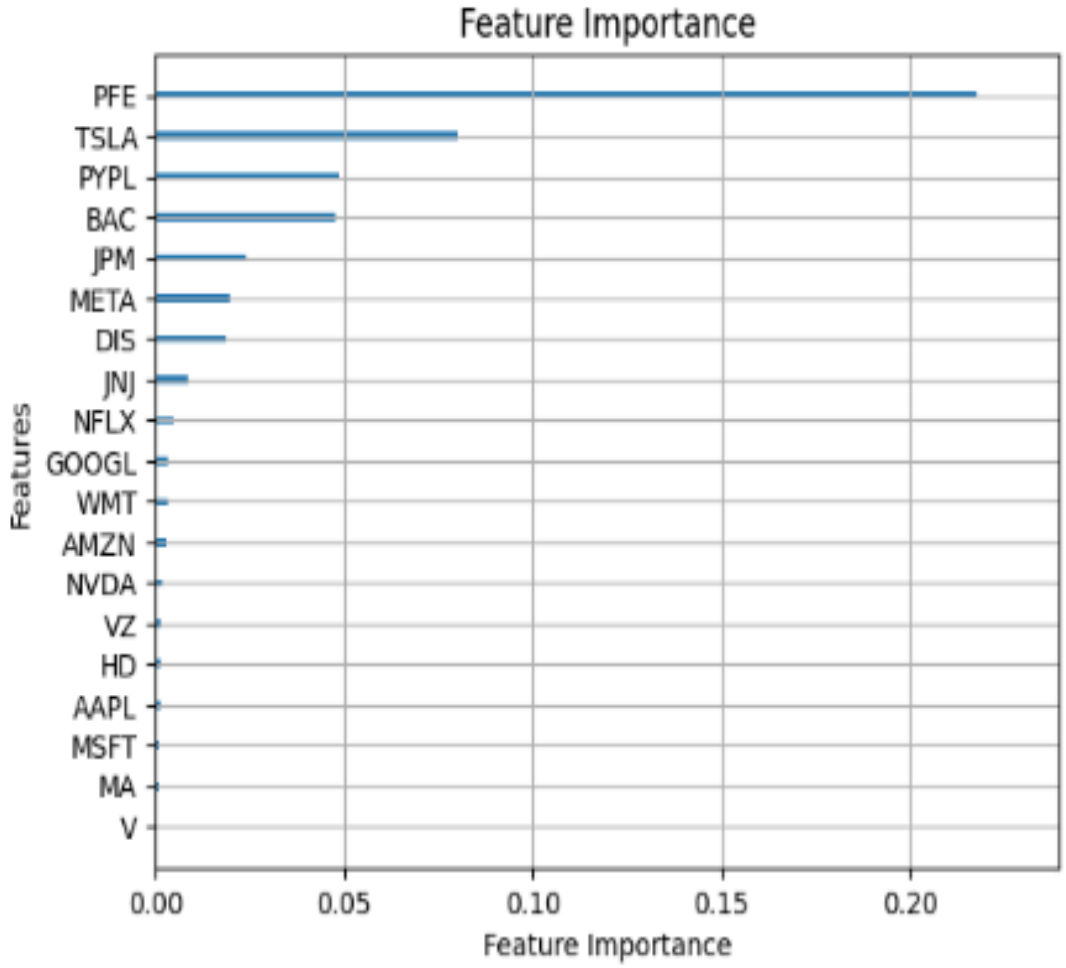


Fig. 23. Feature Importance for the XGB Classifier Model

6.3 Summary of Model results

The results of our model evaluation for the classification algorithms are shown in this section, where we compare the accuracy, precision, recall, F1-score, and performance of the six classifiers on a range of evaluation metrics.

Model:	Rank	Accuracy	F1 -Score	Recall
XGBoostClassifier	1	0.95502645	0.95342466	0.96666666
RandomForestClassifier	2	0.95238095	0.95081967	0.96666666
SVMClassifier	3	0.85449735	0.86352357	0.96666666
Autoencoder Classifier	4	0.87814569		
SOMs Classifier	5	0.54497354		

Table 7. Ranking of Models based on Model Perfomance Metrics

Based on these results, the XGBoost Classifier model emerged as the best machine learning algorithm we can use to select stock pairs to be used in a pairs trading strategy, followed closely by the RandomForestClassifier. Our next step was to tune the XGBoost model to enhance its performance. We used the GridSearchCV function of the sklearn.model_selection Python library to carry out the hyper-parameter tuning. Using a python dictionary, param_grid, we specified a choice of estimators as (100, 200, 300), a choice of learning rate as (0.01, 0.1, 0.3) and a choice of max_depth as (3,5,7). We ran the GridSearch algorithm to determine the best parameters to enhance the model perfomance. The model fitted 5 folds for each of the 27 candidates, for a total of 135 fits. The best parameters found were: learning_rate = 0.1, max_depth = 5 and n_estimator = 300. These hyperparameters are expected to yield the optimal performance for the XGBoost model on the given dataset.

The learning_rate parameter sets the step size at each iteration as it approaches the loss function's minimum. Although it usually takes more iterations, a lower learning rate can lead to better convergence and performance. A tree's maximum depth is specified by the max_depth parameter. The model can identify more intricate patterns in the data by increasing the maximum depth, but doing so also raises the possibility of overfitting. A lower value helps prevent overfitting but may

lead to underfitting. The number of trees in the forest is determined by the n_estimators parameter. The performance of the model is usually enhanced by increasing the number of estimators; however, this also adds to the computational complexity. Although more trees may need more processing power, they can improve generalization. We adopt the tuned model to implement our pairs trading strategy.

Chapter 7: Discussions

7.1 XGBoost Framework

Combining the ideas of machine learning and deep learning, XGBoost is an effective technique for identifying patterns and connections in datasets (Hinton Salakhutdinov, 2006; Xu Wunsch, 2005). In this study, correlated stocks are identified using XGBoost based on price patterns and movements. By making use of XGBoost's capabilities, the strategy seeks to capture intricate non-linear relationships and dependencies between stocks. XGBoost aims to minimize an objective function that combines a loss term and a regularisation term. The objective function is represented as below.

$$Obj(\theta) = \sum_{i=1}^{n} L\left(y_{i,} \hat{y}_i\right) + \sum_{i=1}^{n} \Omega(f_i) \qquad \ldots. (1)$$

$Obj(\theta)$ is the objective function to be minimized.

$L\left(y_{i,} \hat{y}_i\right)$ is the loss term, measuring the error between the true target value $y_{i,}$ and the predicted value $\hat{y}_i$ for each of the sample.

$\Omega(f_i)$is the regularization term for each tree f_i, which prevents overfitting by penalizing complex trees,

θ represents the model parameters.

XGBoost computes the loss function's first derivative (gradient) and second derivative (Hessian) with respect to the predicted values in order to optimize the objective function. This entails computing each sample's gradients and Hessians. Next, XGBoost constructs decision trees in a stepwise manner, beginning with a basic tree and increasing in complexity with each iteration.

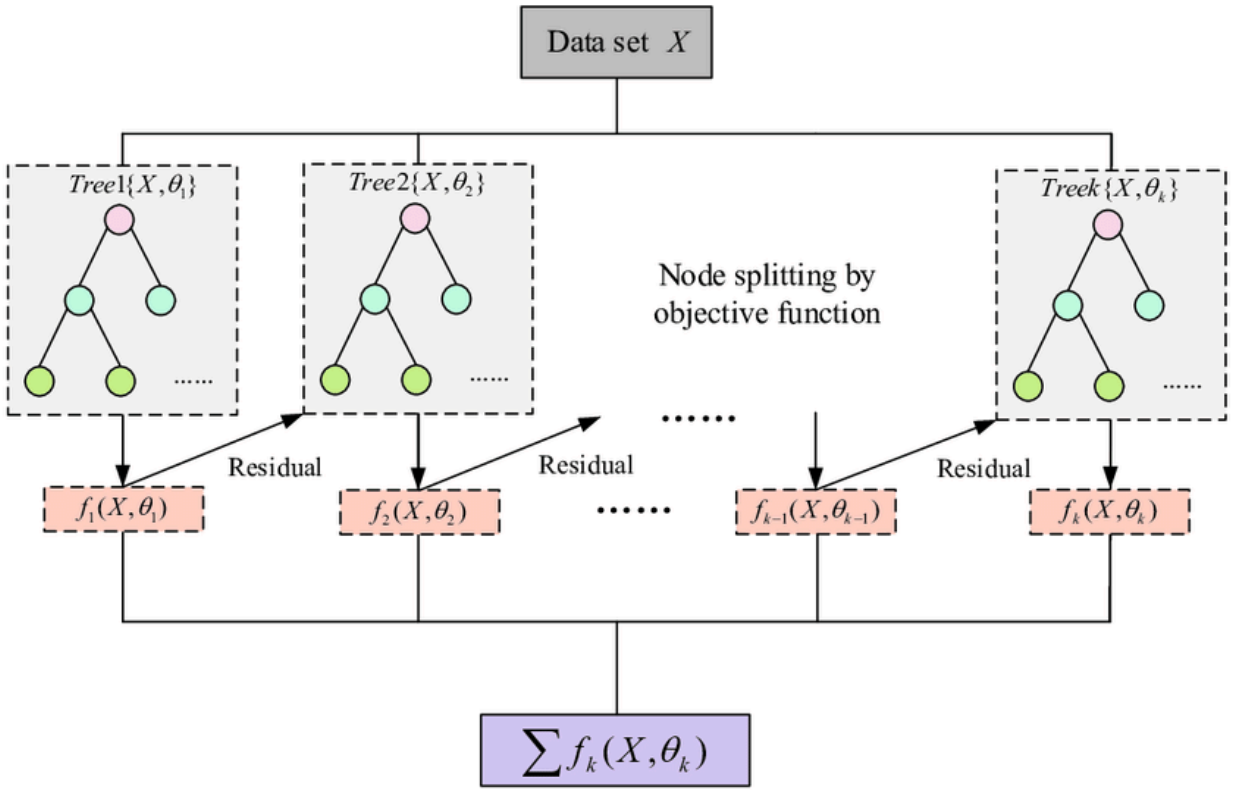


Fig. 24. Tree Boosting Framework

The tree structure is determined based on the gradient and Hessian values. Regularization terms are added to the objective function to control the complexity of individual trees. These regularization terms depend on the structure of the tree, including the number of leaf nodes and the values at each leaf node. The final prediction is obtained by summing the predictions of all the individual trees, with each tree weighted based on its contribution to reducing the objective function. XGBoost often includes a learning rate (η) as a hyperparameter. The learning rate controls the step size during optimization and is multiplied by the predictions of each tree before adding them to the ensemble. To prevent overfitting, XGBoost employs early stopping. Training is halted when the performance on a validation set stops improving. XGBoost provides various hyperparameters related to regularization, such as max_depth, min_child_weight, gamma, and others, which control the structure and complexity of the trees.

Hyperparameter	XGBoost Parameter	Distribution (Range)
Learning Rate	`eta`	$\log\text{-uniform}(10^{-3}, 1.)$
ℓ_1 Regularization	`alpha`	$\log\text{-uniform}(10^{-6}, 2.)$
ℓ_2 Regularization	`lambda`	$\log\text{-uniform}(10^{-6}, 2.)$
Min. Split Loss	`gamma`	$\log\text{-uniform}(10^{-6}, 64.)$
Row Subsample Ratio	`subsample`	$\text{uniform}(0.5, 1.)$
Column Subsample Ratio	`col_subsample`	$\text{uniform}(0.3, 1.)$
Max. Tree Depth	`max_depth`	$\log\text{-randint}(2, 8)$
Boosting Rounds	`num_round`	$\log\text{-randint}(2, 1024)$

Table 8. XGBoost hyper-parameters

7.2 Tuned Model Results

When we run our tuned XGBoost Classifier model on the test dataset, we are able to get a set of stock pairs which we then use to implement our pairs trading strategy. The results below show the trading performance of the top 10 stock pairs from our model.

```
Top 10 Profitable Pairs:
Pair: ('AMZN', 'NFLX'), Profit: 2116990.372001122
Pair: ('META', 'NFLX'), Profit: 1719787.7236810972
Pair: ('NVDA', 'MA'), Profit: 1037428.4852115264
Pair: ('AMZN', 'MA'), Profit: 943150.3777594096
Pair: ('V', 'MA'), Profit: 744842.4102323424
Pair: ('TSLA', 'HD'), Profit: 714304.8293853778
Pair: ('AMZN', 'HD'), Profit: 713549.9375674399
Pair: ('AAPL', 'MA'), Profit: 623176.9791828331
Pair: ('V', 'HD'), Profit: 590884.850449815
Pair: ('AAPL', 'HD'), Profit: 585288.8223078081
```

The results indicate the profitability of executing a pairs trading strategy using the top 10 most profitable pairs identified by the XGBoost model.

Pair: ('AMZN', 'NFLX'), Profit: 2,116,990.37:

This pair consists of Amazon (AMZN) and Netflix (NFLX). The trading strategy executed on this pair yielded a profit of approximately $2,116,990. This indicates a strong potential for profit

when trading between Amazon and Netflix stocks.

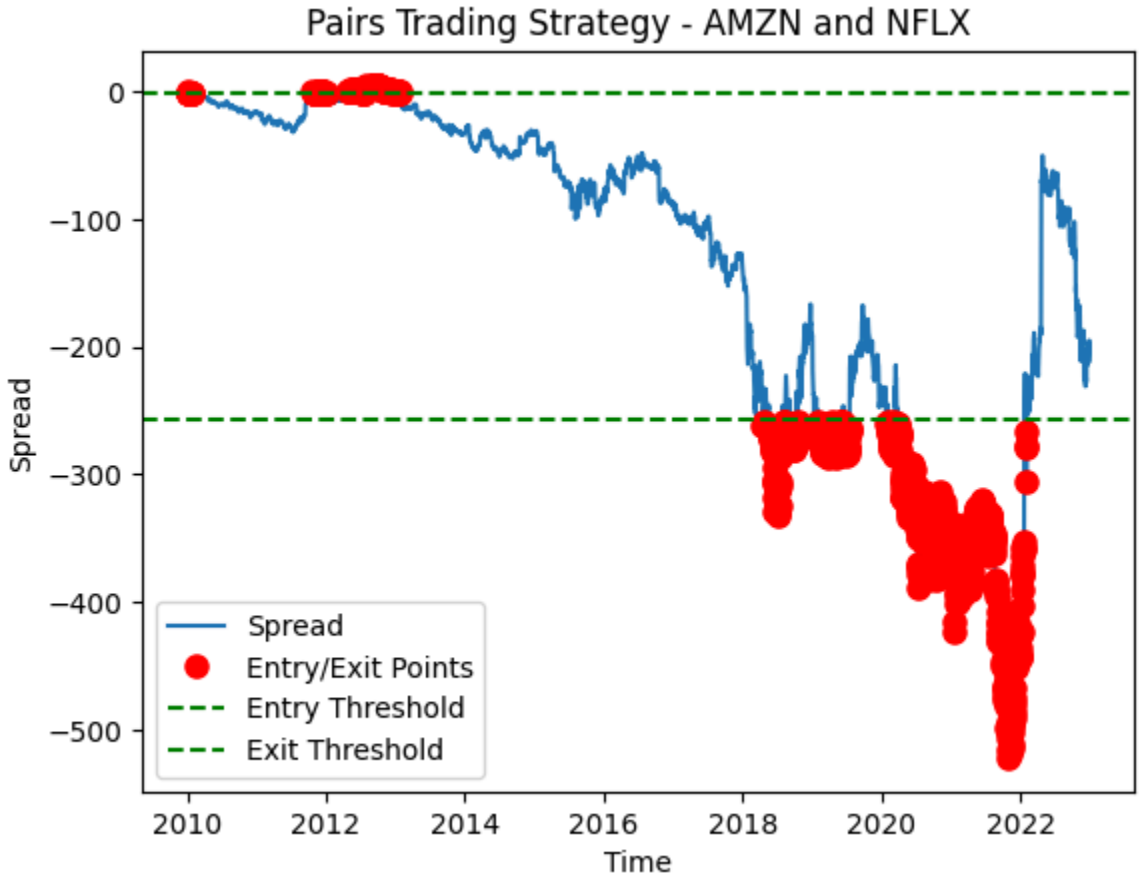


Fig. 25. Pairs Trading Strategy - AMZN and NFLX

Pair: ('META', 'NFLX'), Profit: 1,719,787.72:

This pair consists of Meta Platforms Inc. (formerly Facebook) (META) and Netflix (NFLX). The trading strategy executed on this pair yielded a profit of approximately $1,719,788. Trading between these two stocks resulted in significant profitability.

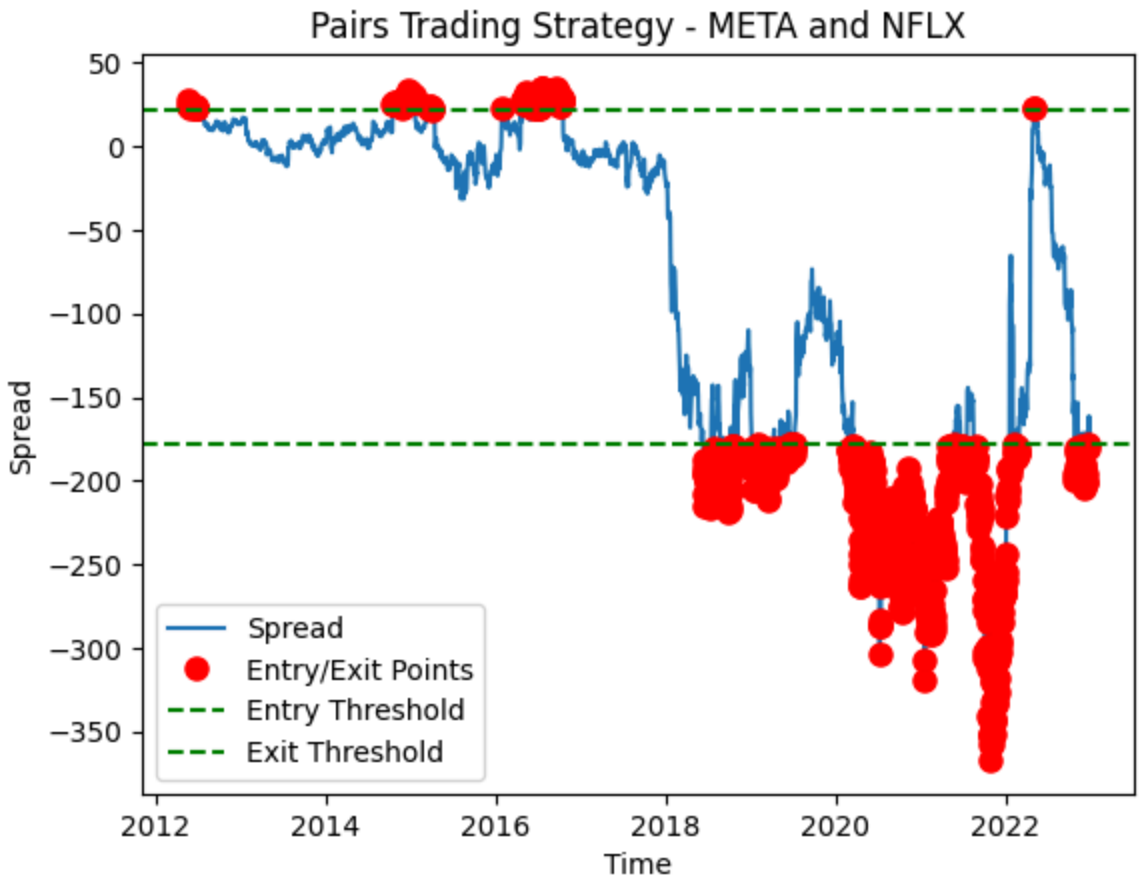


Fig. 26. Pairs Trading Strategy - META and NFLX

Pair: ('NVDA', 'MA'), Profit: 1,037,428.48:

This pair consists of NVIDIA Corporation (NVDA) and Mastercard (MA). The trading strategy executed on this pair yielded a profit of approximately $1,037,428. Both NVDA and MA showed significant profit potential when traded as a pair.

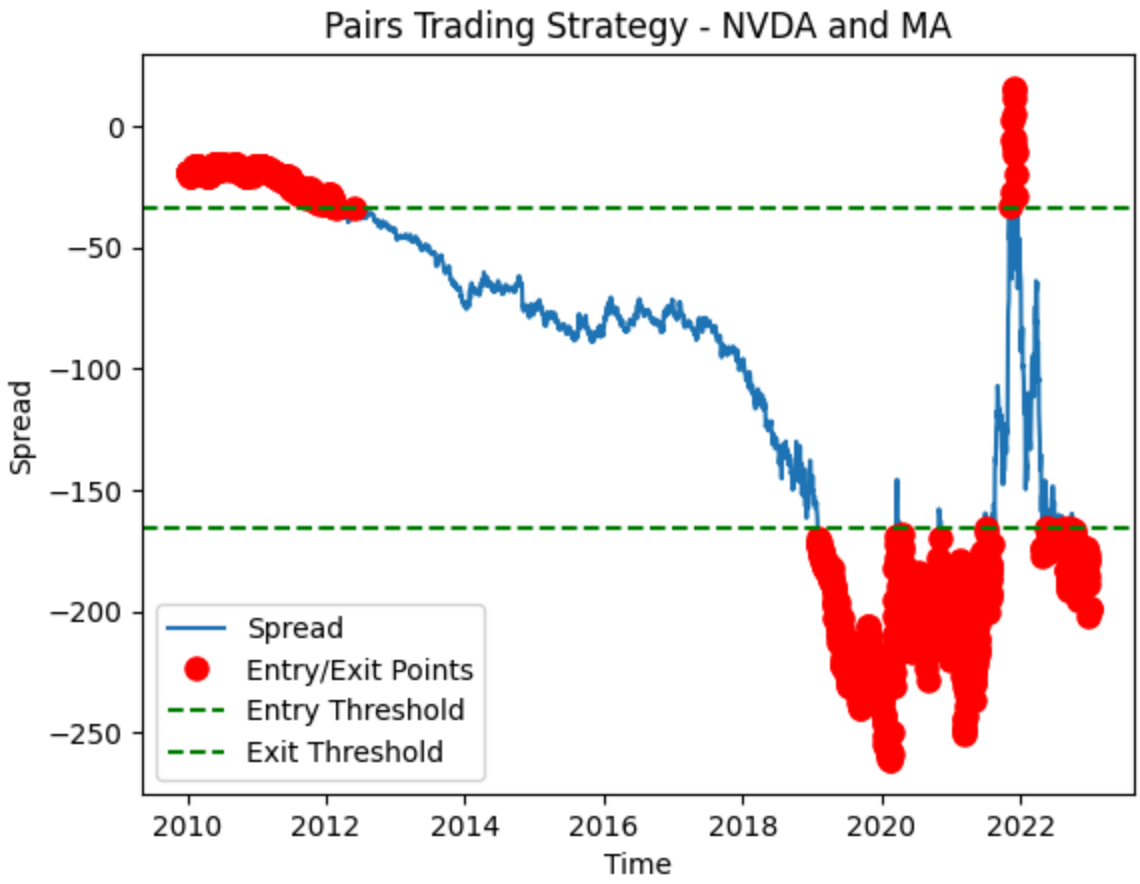


Fig. 27. Pairs Trading Strategy - NVDA and MA

Pair: ('AMZN', 'MA'), Profit: 943,150.37:

This pair consists of Amazon (AMZN) and Mastercard (MA). The trading strategy executed on this pair yielded a profit of approximately $943,150.

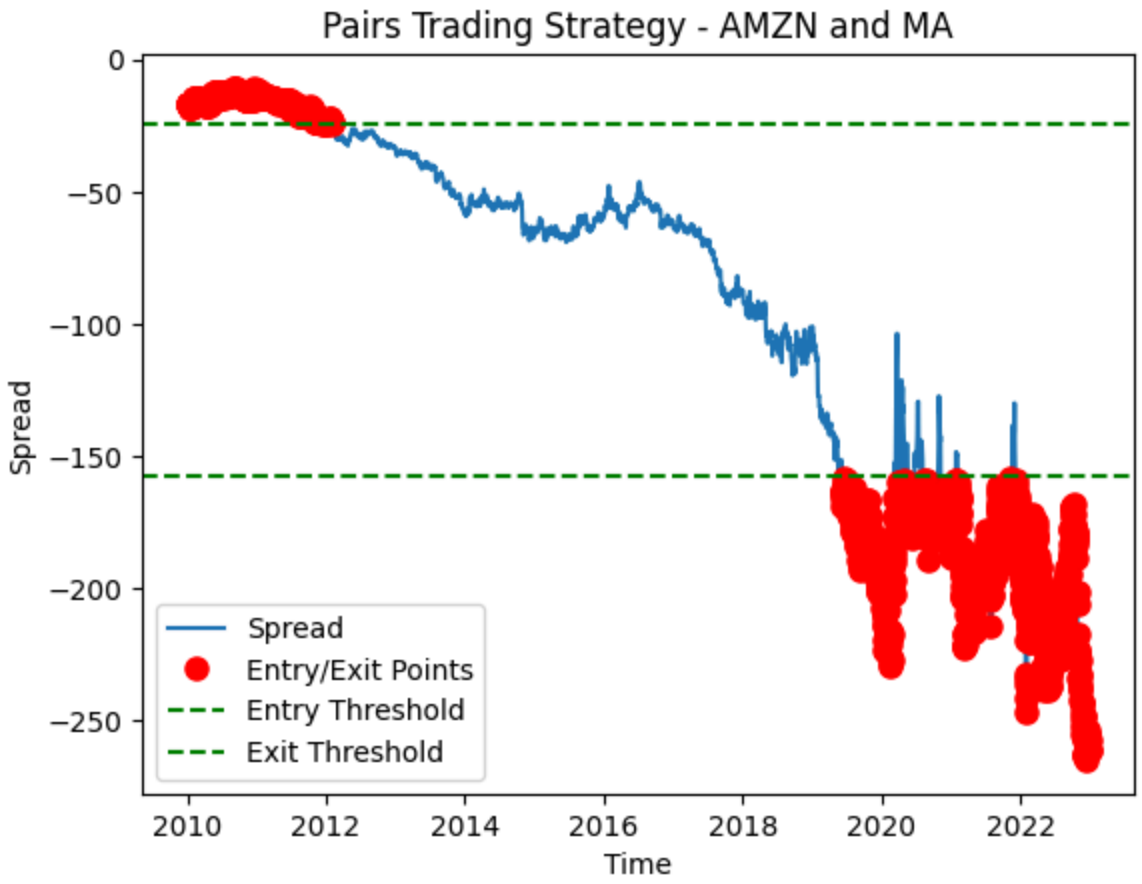


Fig. 28. Pairs Trading Strategy - AMZN and MA

Pair: ('V', 'MA'), Profit: 744,842.41:

This pair consists of Visa Inc. (V) and Mastercard (MA). The trading strategy executed on this pair yielded a profit of approximately $744,842. Visa and Mastercard stocks showed potential for profit when traded together.

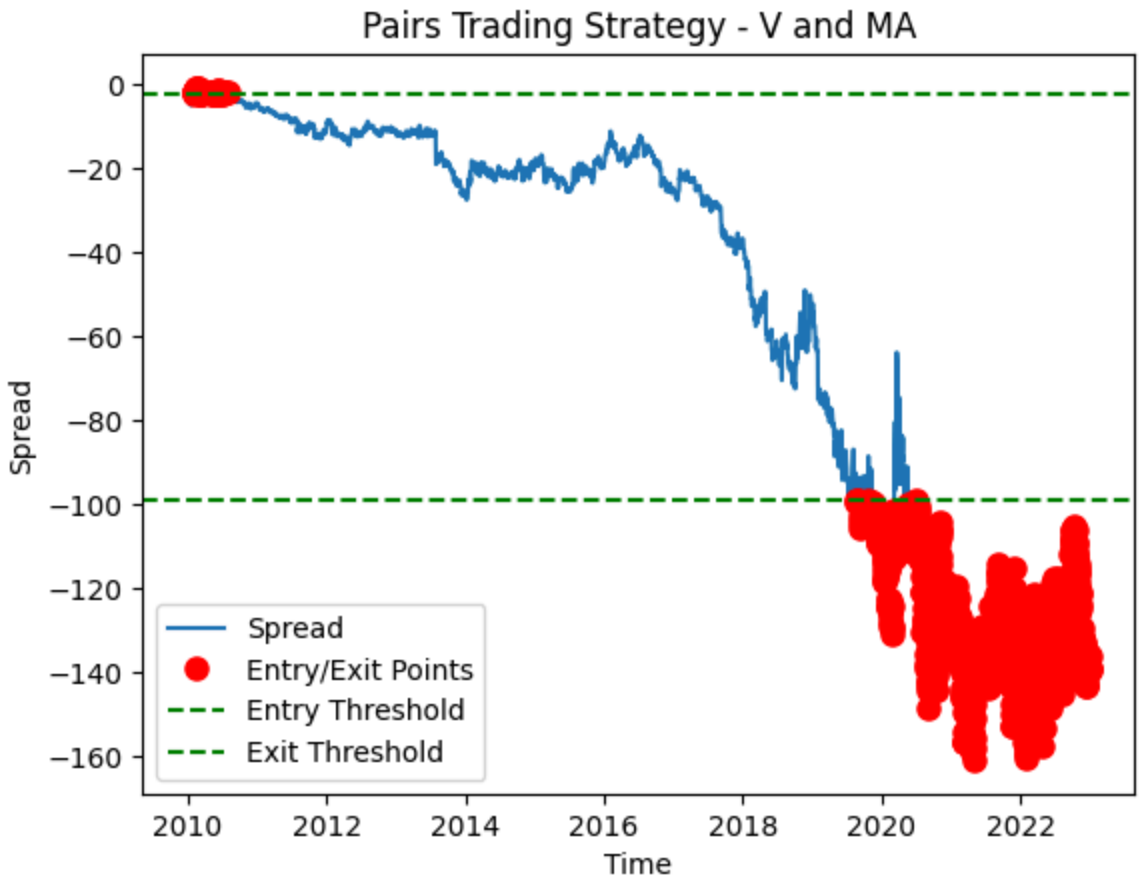


Fig. 29. Pairs Trading Strategy - V and MA

Pair: ('TSLA', 'HD'), Profit: 714,304.82:

This pair consists of Tesla (TSLA) and The Home Depot (HD). The trading strategy executed on this pair yielded a profit of approximately $714,305.

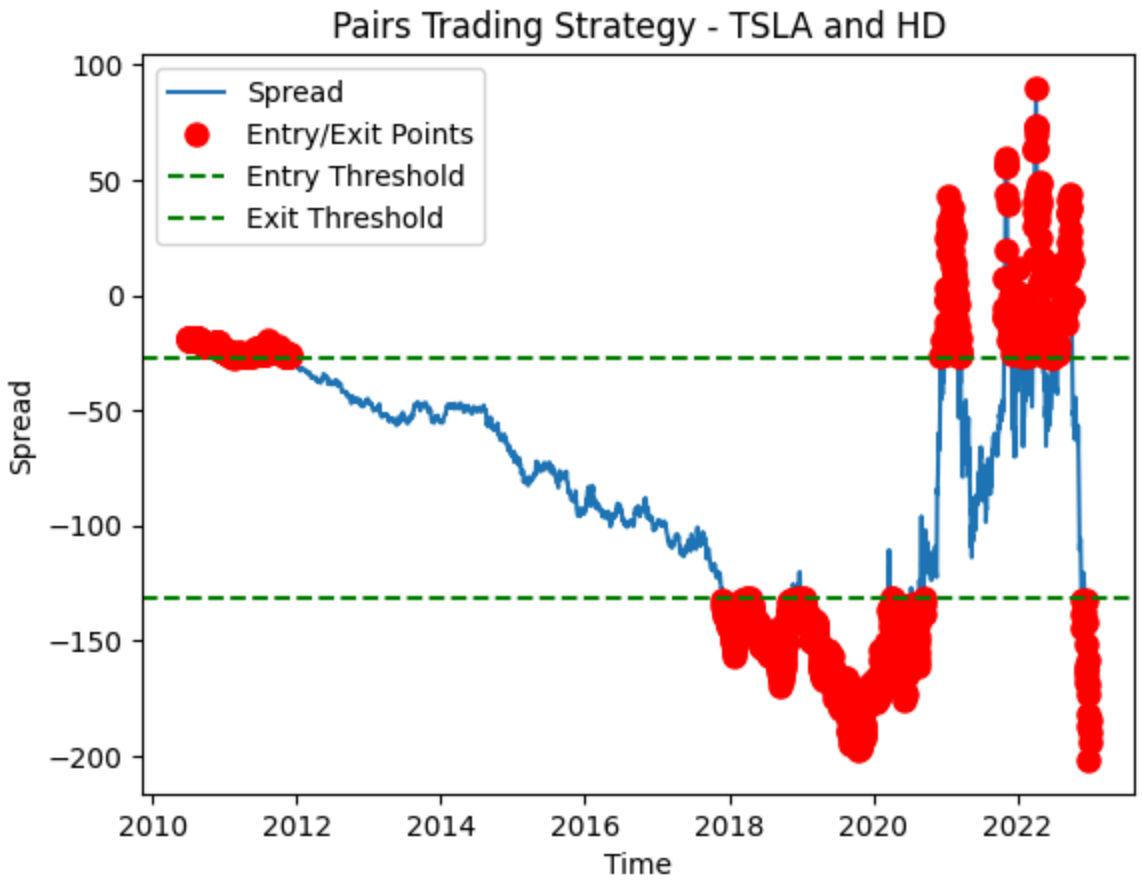


Fig. 30. Pairs Trading Strategy - TSLA and HD

Pair: ('AMZN', 'HD'), Profit: 713,549.93:

This pair consists of Amazon (AMZN) and The Home Depot (HD). The trading strategy executed on this pair yielded a profit of approximately $713,550.

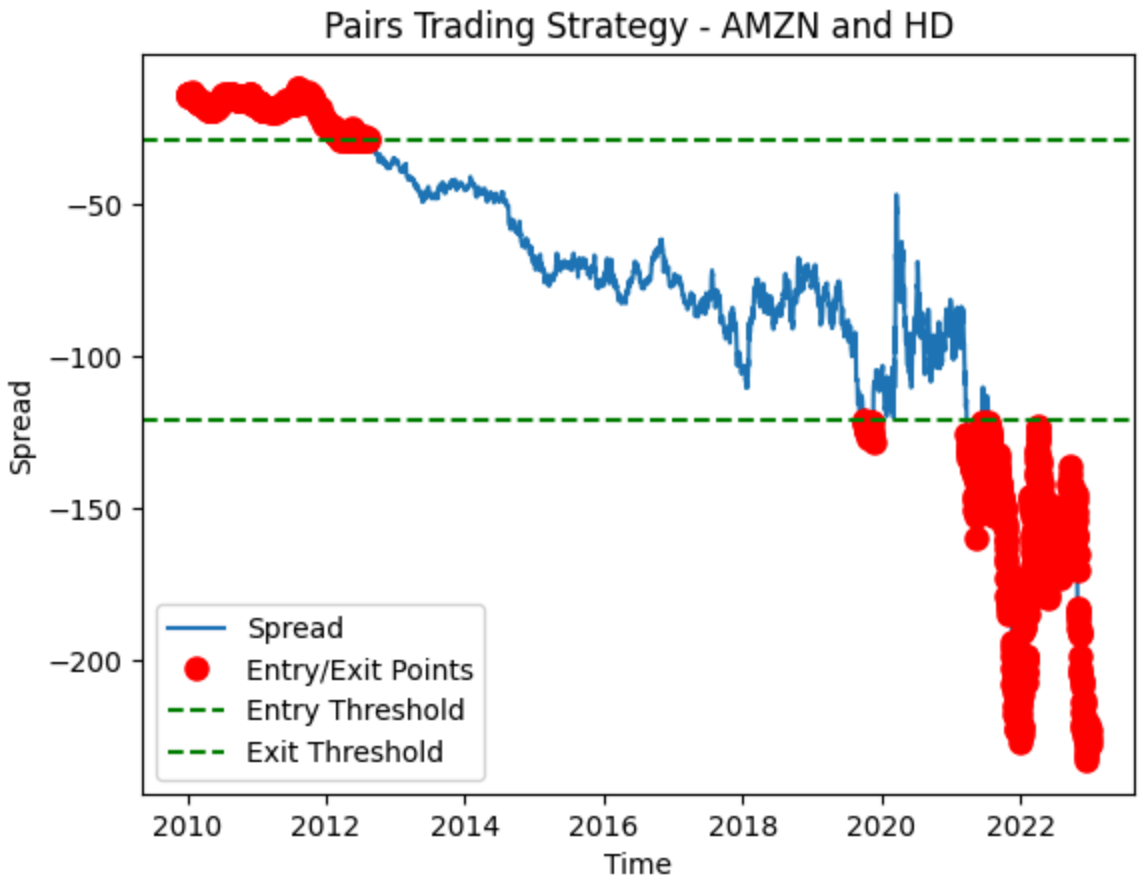


Fig. 31. Pairs Trading Strategy - AMZN and HD

Pair: ('AAPL', 'MA'), Profit: 623,176.97:

This pair consists of Apple (AAPL) and Mastercard (MA). The trading strategy executed on this pair yielded a profit of approximately $623,177.

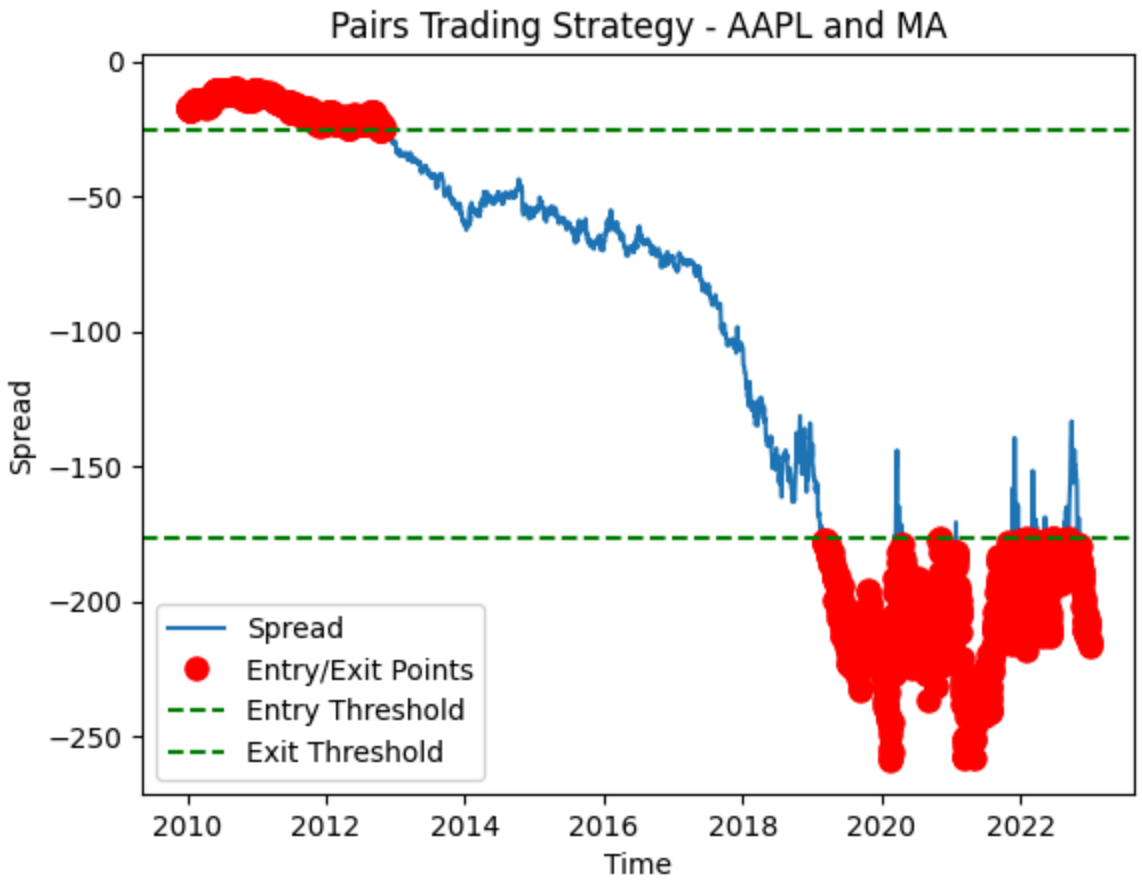


Fig. 32. Pairs Trading Strategy - AAPL and MA

Pair: ('V', 'HD'), Profit: 590,884.85:

This pair consists of Visa Inc. (V) and The Home Depot (HD). The trading strategy executed on this pair yielded a profit of approximately $590,885. Visa and The Home Depot stocks showed potential for profit when traded together.

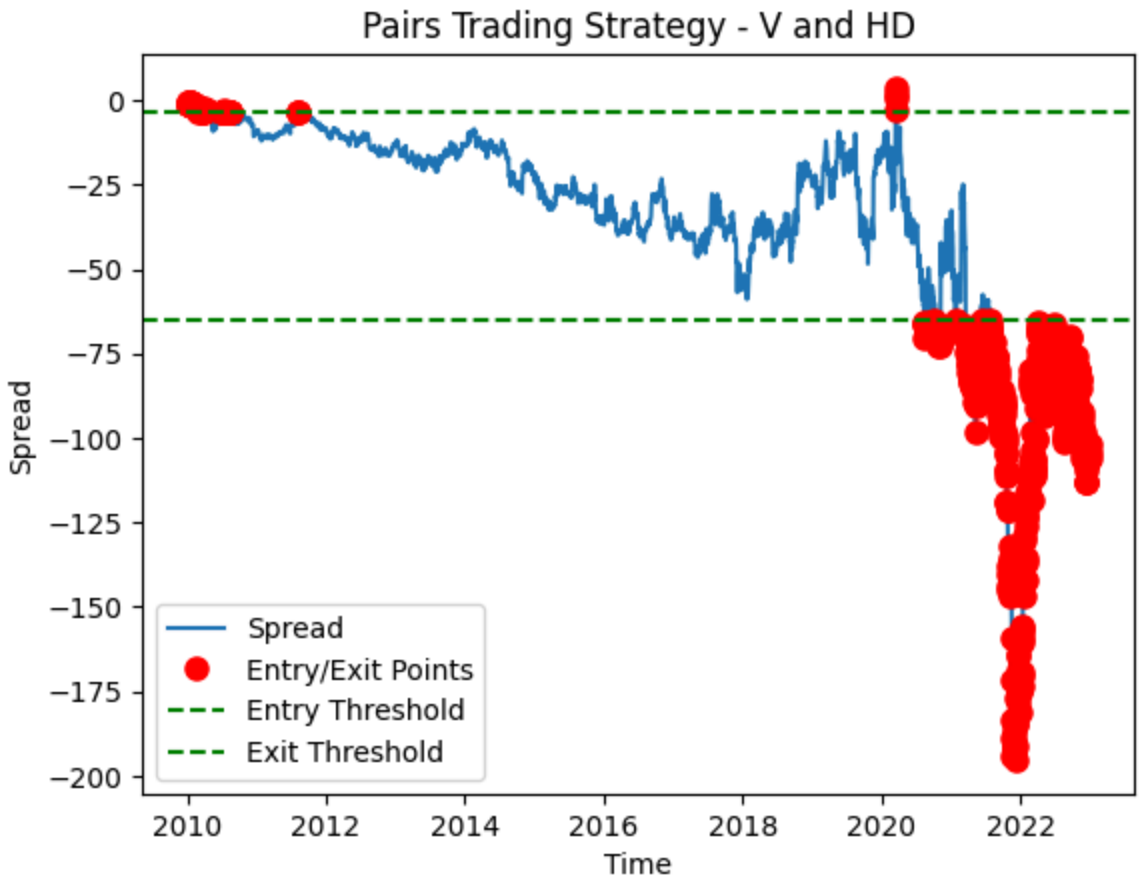


Fig. 33. Pairs Trading Strategy - V and HD

Pair: ('AAPL', 'HD'), Profit: 585,288.82:

This pair consists of Apple (AAPL) and The Home Depot (HD). The trading strategy executed on this pair yielded a profit of approximately $585,289. Trading between Apple and The Home Depot resulted in significant profitability.

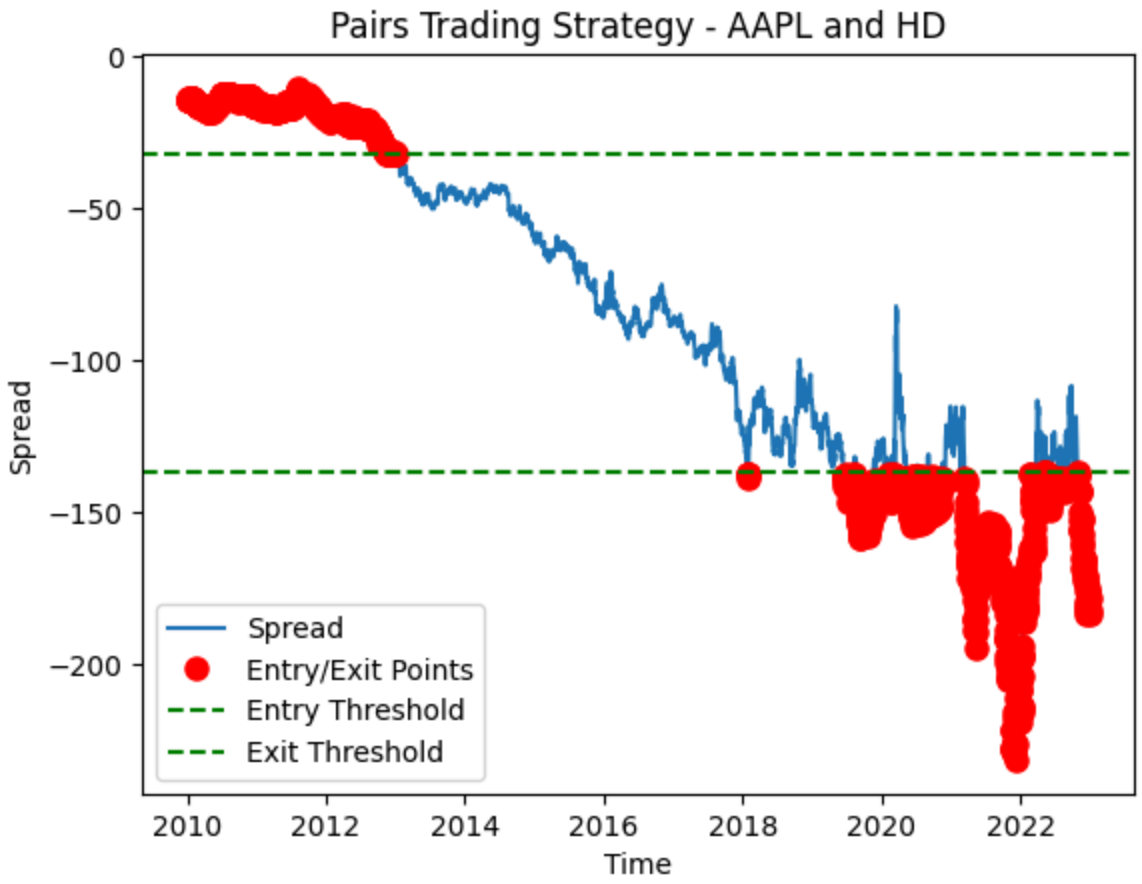


Fig. 34. Pairs Trading Strategy - AAPL and HD

These results show that pairs trading using the identified pairs from the XGBoost model can be highly profitable, with substantial gains achievable within a given trading period.

Further analyzing the performance of a trader, starting out with initial investment capital of 100,000 dollars and who undertook trades in the top 10 stock pairs as a portfolio, advised by the XG Boost Classifier model, we see that he would have the below performance metrics:

Performance Metric	Return
Return on Investment	9,789%
Annualized Return	97.9%
Win Rate	1
Maximum Drawdown	5.85

Volatility	4.98

Table 9. XGBoost Portfolio Profitability Metrics

Return on Investment, or ROI, measures how profitable a trader's investments were in comparison to their initial outlay of funds. The ROI in this instance is roughly 9,789.40%, which denotes a high rate of return on investment. The average rate of return over a one-year period is measured by the annualized return. A figure of 97.89% indicates a high yearly average return on investment. The percentage of profitable trades in relation to all trades is known as the win rate. A win rate of 1.0 is an exceptional result, meaning that every trade was profitable. The reason for this is that the trader used the XGBoost Classfier model's recommended portfolio of the top 10 profitable stock pairs, backed by the trading thresholds used to carry out the pairs trading strategy. Maximum Drawdown: The biggest decrease in portfolio value from its highest point to its lowest is measured by the maximum drawdown. A maximum loss that was sustained during the trading period is represented by a value of roughly 5.85%. Volatility: The degree of variance in a trader's returns is measured by volatility. A volatility of roughly 4.98% indicates that returns will fluctuate somewhat during the trading session. The trader achieved exceptional performance across all metrics with high returns, a perfect win rate, and relatively low drawdown and volatility while using this model and trading strategy.

7.3 Tuned Model Results applied to the Validation Period Data

When we applied the model to the validation data set (January 1st, 2023 to December 31st, 2023) and used the same pairs trading strategy, the profitability of the portfolio dropped. We obtained below results:

```
Top 10 Profitable Pairs:
Pair: ('META', 'NVDA'), Profit: 145351.96518822553
Pair: ('AAPL', 'NVDA'), Profit: 123975.65823431083
Pair: ('AMZN', 'NVDA'), Profit: 123876.8400297506
Pair: ('MSFT', 'NVDA'), Profit: 120955.68234253721
Pair: ('GOOGL', 'NVDA'), Profit: 119606.65446018765
Pair: ('TSLA', 'NVDA'), Profit: 118004.3604878966
Pair: ('AAPL', 'META'), Profit: 100080.35001082631
Pair: ('AMZN', 'META'), Profit: 93954.73825645735
```

Pair: ('MSFT', 'META'), Profit: 87462.86927748861

Pair: ('GOOGL', 'META'), Profit: 81447.3394645307

Historical Total Profit	9,789,404.78
Validation Total Profit	1,114,716.46
Historical Sharpe Ratio	1.967
Validation Sharpe Ratio	5.899

Table 10. XGBoost Portfolio Profitability Metrics - Out of Sample

Whereas the historical total profit is significantly higher than the validation total profit. The Historical Sharpe Ratio is 1.967 while the Validation Sharpe Ratio is5.899. The Sharpe ratio measures the risk-adjusted return of an investment strategy. A higher Sharpe ratio indicates better risk-adjusted returns. In this case, the validation Sharpe ratio is substantially higher than the historical Sharpe ratio, implying that, relative to the risk taken, the strategy performed exceptionally well during the validation period compared to the historical period. The Sharpe ratio suggests that the strategy's performance in the validation period was more efficient in terms of risk-adjusted returns, suggesting that the strategy is still robust and adaptive to new market conditions. To validate the efficacy of the selected XGBoost model, we applied the trading strategy to the stock pairs selected by the 2nd and 3rd runner up models and obtained the following results:

Top 10 Profitable Pairs Using XGBoost model:		**Top 10 Profitable Pairs Using Random Forest:**		**Top 10 Profitable Pairs Using SVM:**	
Pair: ('META', 'NVDA'),	145,351.96	Pair: ('MSFT', 'NVDA'),	60,085.04	Pair: ('BAC', 'HD'),	23,593.96
Pair: ('AAPL', 'NVDA'),	123,975.66	Pair: ('GOOGL', 'NVDA'),	58,860.20	Pair: ('BAC', 'XOM'),	19,248.49

Pair: ('AMZN', 'NVDA'),	123,876.84	Pair: ('BAC', 'HD'),	45,808.84	Pair: ('JPM', 'XOM'),	10,606.45
Pair: ('MSFT', 'NVDA'),	120,955.67	Pair: ('BAC', 'XOM'),	45,618.15	Pair: ('AAPL', 'MSFT'),	0.00
Pair: ('GOOGL', 'NVDA'),	119,606.65	Pair: ('AAPL', 'NVDA'),	44,041.78	Pair: ('AAPL', 'NVDA'),	0.00
Pair: ('TSLA', 'NVDA'),	118,004.36	Pair: ('GOOGL', 'META'),	32,266.67	Pair: ('MSFT', 'NVDA'),	0.00
Pair: ('AAPL', 'META'),	100,080.35	Pair: ('AAPL', 'MSFT'),	25,555.90	Pair: ('GOOGL', 'META'),	0.00
Pair: ('AMZN', 'META'),	93,954.74	Pair: ('GOOGL', 'HD'),	22,922.44	Pair: ('GOOGL', 'NVDA'),	0.00
Pair: ('MSFT', 'META'),	87,462.87	Pair: ('MSFT', 'HD'),	19,429.81	Pair: ('TSLA', 'NFLX'),	-3,073.16
Pair: ('GOOGL', 'META'),	81,447.34	Pair: ('AAPL', 'HD'),	12,909.87	Pair: ('AAPL', 'HD')	-7,328.09
Total	1,114,716.44		367,498.70		43,047.65

Table 11. Comparison of Trading Portfolio Outcomes Based on the Top Three Models

The analysis of the top 10 profitable pairs using the three different models reveals interesting insights into their respective performances. The XGBoost model stands out prominently, demonstrating superior predictive power and yielding a total profit of $1,114,716.44. Notably, pairs involving NVDA emerged as the most profitable, with stocks like ('META', 'NVDA'), ('AAPL', 'NVDA'), and ('AMZN', 'NVDA') yielding substantial profits. This indicates the model's effectiveness in identifying NVDA's behavior relative to other stocks, highlighting its potential for accurately predicting market movements. In contrast, the Random Forest model, while able to identify some profitable pairs such as ('BAC', 'HD') and ('BAC', 'XOM'), lagged significantly behind XGBoost, with a total profit of $367,498.70. Despite demonstrating the ability to uncover unique opportunities, the model struggled to match the profitability of

XGBoost. The SVM model exhibited the lowest profitability among the three, totaling $43,047.65. While it identified profitable pairs involving tech stocks like MSFT and NVDA, its overall performance fell short compared to XGBoost and Random Forest. This suggests potential limitations in capturing subtle stock relationships compared to the other two models. The XGBoost model portfolio demonstrates high profitability, with the top pairs yielding substantial profits. It indicates strong predictive power in identifying profitable stock pairs.

Chapter 8: Conclusion

Pairs trading, a market-neutral strategy, aims to capitalize on price divergences between highly correlated stocks, by taking opposite positions in a stock pair, i.e. a long position in one stock and a short position in the other, or vice versa. The problem statement underscored the importance of accurately identifying correlated stock pairs to capitalize on price divergences effectively. While traditional statistical and deep learning methods have historically provided techniques of selecting stock pairs to be used in pairs trading strategies, their limitations in capturing complex nonlinear dependencies and subtle correlations in financial datasets have prompted the exploration of advanced techniques with capabilities of addressing these shortcomings. The data and data collection methods employed in this research effort played a pivotal role in facilitating a comprehensive analysis of pairs trading strategies. Leveraging daily price data sourced from the Yahoo Finance website for a sample of 20 stocks listed on the S&P 500 index, spanning from January 1st, 2010, to December 31st, 2022, ensured a rich and extensive dataset capturing over a decade of historical market activity. Additionally, the validation period from January 1st, 2023, to December 31st, 2023, provided valuable insights into the model's performance in a more recent market environment.

This research delved into enhancing pairs trading strategies by leveraging the Extreme Gradient Boosting (XGBoost) algorithm. Initially, we hypothesized that XGBoost could outperform traditional methods and other machine learning algorithms in identifying correlated stock pairs. Our investigation indeed revealed that the tuned XGBoost model excelled, showcasing a remarkable accuracy of 95.50% and a precision of 95.34% in identifying profitable stock pairs. XGBoost's functionality played a pivotal role in achieving these impressive results. By minimizing an objective function combining loss and regularization terms, XGBoost effectively optimized model performance. Specifically, we utilized the GridSearchCV function to fine-tune the model's hyperparameters. This included setting a learning rate of 0.1, a maximum depth of 5, and the number of estimators to 300. The computation of gradients and Hessians facilitated the construction of decision trees, enabling the model to capture intricate non-linear relationships in stock price data.The application of XGBoost was grounded in its ability to capture intricate non-linear relationships and dependencies in stock price data, as demonstrated by its superior

performance in comparison to Random Forest and SVM classifiers, the two runners up models considered in the study.

Further analysis of the trading performance of the stock pairs selected using the XGBoost model uncovered substantial profitability. For instance, the top 10 profitable pairs yielded profits ranging from $585,288.82 to $2,116,990.37, when the strategy was applied to the 12 year training period and assuming an initial investment capital of $100,000; resulting in a remarkable return on investment (ROI) of approximately 9,789.40% over the period. This exceptional ROI underscores the efficacy of the XGBoost model in identifying lucrative trading opportunities. Additionally, the model demonstrated a perfect win rate, implying that every trade recommended by the model was profitable. This stellar performance, coupled with a relatively low maximum drawdown of 5.85% and volatility of 4.98%, indicates the robustness and stability of the pairs trading strategy advised by the XGBoost model. However, when we applied the model to a validation dataset, the profitability decreased, though the Sharpe ratio increased significantly to 5.899. This suggests that the model maintained its adaptability and efficiency in risk-adjusted returns even in new market conditions. Moreover, the analysis of profitable pairs during the validation period revealed insights into market dynamics, with pairs involving NVDA emerging as the most profitable. Despite the decline in profitability during validation, the XGBoost model consistently outperformed Random Forest and SVM models, underscoring its superiority in identifying profitable stock pairs.

Despite the promising results and contributions to the field, this research effort is not without its limitations. Firstly, while XGBoost demonstrated superior performance in identifying correlated stock pairs for pairs trading, it's important to acknowledge that no model is perfect. The XGBoost algorithm, like any other machine learning model, has its limitations. One notable limitation is its susceptibility to overfitting, especially when dealing with noisy or sparse datasets. While techniques such as regularization can help mitigate this issue, it remains a challenge, particularly in financial datasets characterized by high dimensionality and volatility. Additionally, the performance of the XGBoost model is highly dependent on the quality and quantity of input features. Inadequate or irrelevant features may lead to suboptimal results. Furthermore, the tuning of hyperparameters, while essential for optimizing model performance, can be time-consuming and computationally intensive, limiting the scalability of the approach.

Lastly, the research focused solely on comparing XGBoost with traditional statistical and deep learning methods, leaving room for exploring other machine learning algorithms or ensemble techniques that may offer further improvements. Overall, while XGBoost shows great promise in pairs trading, acknowledging and addressing these limitations is crucial for future research and practical implementation.

In conclusion, this research provides compelling evidence for the effectiveness of the tuned XGBoost algorithm in identifying correlated stock pairs for pairs trading strategies. The model's remarkable accuracy, high ROI, perfect win rate, and low volatility make it a valuable tool for traders seeking to exploit market inefficiencies and generate profits. While there may be fluctuations in profitability when applied to unseen data, the model's robustness and adaptability ensure consistent risk-adjusted returns. Moving forward, further research could focus on refining the model's performance in diverse market conditions and integrating additional data sources to enhance predictive capabilities. Overall, the findings of this research offer actionable insights for practitioners and researchers interested in developing and implementing pairs trading strategies, leveraging the power of machine learning algorithms like XGBoost.

References

Ahn, Jung Min and Kim, Jungwook and Kim, Hongtae and Kim, Kyunghyun, Ensemble Machine Learning of Gradient Boosting(XGboost, LightGBM, CatBoost) And Attention-Based CNN-LSTM for Harmful Algal Blooms Forecasting. Available at SSRN: https://ssrn.com/abstract=4434784 or http://dx.doi.org/10.2139/ssrn.4434784

Andersen, H.W., & Tronvoll, H. (2018). Statistical arbitrage trading with implementation of machine learning : an empirical analysis of pairs trading on the Norwegian stock market.

Anju, & Hazarika, A.V. (2017). Extreme Gradient Boosting using Squared Logistics Loss function.

Anju, & Sharma, N. (2018). Survey of Boosting Algorithms for Big Data Applications. International journal of engineering research and technology, 5.

Baek, S., Glambosky, M., Oh, S.H., & Lee, J.W. (2020). Machine Learning and Algorithmic Pairs Trading in Futures Markets. Sustainability.

Brunetti, M., & De Luca, R. (2020). Pre-selection in Cointegration-based Pairs Trading. CEIS: Centre for Economic & International Studies Working Paper Series.

Breiman, L., & Rice, J. (2003). Random Forests: Finding Quasars.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. Journal of Political Economy, 81(3), 637–654.

Brockwell, P. J., Davis, R. A. (2016). Introduction to Time Series and Forecasting (3rd ed.). Springer.

Chan, E. P. (2013). Algorithmic trading: winning strategies and their rationale. John Wiley & Sons.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Caldeira, J.F., & Moura, G.V. (2013). Selection of a Portfolio of Pairs Based on Cointegration: A Statistical Arbitrage Strategy. Advanced Risk & Portfolio Management® Research Paper Series.

Chrétien, S., Lohvithee, M., Sun, W., & Soleimani, M. (2020). Efficient Hyper-Parameter Selection in Total Variation-Penalised XCT Reconstruction Using Freund and Shapire's Hedge Approach. Mathematics.

Deshpande, A., & Barmish, B.R. (2016). A general framework for pairs trading with a control-theoretic point of view. 2016 IEEE Conference on Control Applications (CCA), 761-766.

Elliott, R.J., Van Der Hoek *, J., & Malcolm, W.P. (2005). Pairs trading. Quantitative Finance, 5, 271 - 276.

Engle, R. F.; Granger, C. W. (1987). Co-integration and error correction: representation, estimation, and testing. Econometrica, 55 (2), 251-276.

Fafalios, S., Charonyktakis, P., & Tsamardinos, I. (2020). Gradient Boosting Trees.

Fama, E. F.; French, K. R. (2004). The Capital Asset Pricing Model: Theory and Evidence. Journal of Economic Perspectives, 18(3), 25–46. https://doi.org/10.1257/0895330042162386

Fama, E. F.; French, K. R. (1993). Common risk factors in the returns on stocks and bonds. Journal of Financial Economics, 33(1), 3–56.

Fil, M., & Kristoufek, L. (2020). Pairs Trading in Cryptocurrency Markets. IEEE Access, 8, 172644-172651.

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29, 1189-1232.

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

Gatev, E., Goetzmann, W. N., and Rouwenhorst, K. G. (1999). Pairs trading: Performance of a relative-value arbitrage rule. Review of Financial Studies, 12(5), 853–891.

Ghatak, A. (2011). PAIR TRADING STRATEGY IN INDIAN CAPITAL MARKET: A COINTEGRATION APPROACH. TIJ's Research Journal of Social Science & Management - RJSSM, 1.

Guo, X., and Zhu, H. (2016). The pairs trading strategy is based on the distance between two stocks. Journal of Applied Statistics, 43(14), 2562–2579.

Gurrib, I. (2014). A pairs trading strategy based on the time-varying copula model. International Journal of Economics and Finance, 6(7), 16–25

Hilton, J. (2009). The Origins of the Global Financial Crisis. Harvard Business School Working Papers, (10-061).

Hong, S.G., & Hwang, S. (2022). In search of pairs using firm fundamentals: is pairs trading profitable? The European Journal of Finance, 29, 508 - 526.

Huang, M., Chen, C., Lin, W., Ke, S., & Tsai, C. (2017). SVM and SVM Ensembles in Breast Cancer Prediction. PLoS ONE, 12.

Huck, N., & Afawubo, K. (2015). Pairs trading and selection methods: is cointegration superior? Applied Economics, 47, 599 - 613.

Huck, N. (2015). Pairs trading: does volatility timing matter? Applied Economics, 47, 6239 - 6256.

Jæger, M.H. (2016). Dynamic Cointegration Based Pairs Trading.

Jacobsen, B., and Zhang, C. (2013). Stock return predictability and variance risk premia: statistical inference and international evidence. Journal of Financial and Quantitative Analysis, 48 (5), 1553–1580.

Jirapongpan, R., & Phumchusri, N. (2020). Prediction of the Profitability of Pairs Trading Strategy Using Machine Learning. 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA), 1025-1030.

Kapoor, J. R., Dlabay, L. R., and Hughes, R. J. (2017). Focus on Personal Finance. McGraw-Hill Education.

Kapoor, S., & Perrone, V. (2021). A Simple and Fast Baseline for Tuning Large XGBoost Models. ArXiv, abs/2111.06924.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43(1), 59–69.

Kohonen, T. (1995). Self-Organizing Maps. Springer Series in Information Sciences.

Kohonen, T. (2004). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43, 59-69.

Kumar, A., and Avinash, K. (2021). Pairs trading: An empirical analysis using the VECM model. International Journal of Business Analytics and Intelligence, 8(1), 1–16Y.,

Kuo, W., Chang, W., Dai, T., Chen, Y., & Chang, H. (2022). Improving Pairs Trading Strategies Using Two-Stage Deep Learning Methods and Analyses of Time (In)variant Inputs for Trading Performance. IEEE Access, 10, 97030-97046.

Lee, J. H. and Avellaneda (2010). Statistical arbitrage in the US equities market, Quantitative Finance, 10(7), 761–782.

Lee, J. H. and Avellaneda (2010). Statistical arbitrage in the US equities market, Quantitative Finance, 10(7), 761–782.

Leung, T., & Yan, R. (2018). Optimal Dynamic Pairs Trading of Futures Under a Two-Factor Mean-Reverting Model. Derivatives eJournal.

Lintilhac, P.S., & Tourin, A. (2017). Model-based pairs trading in the bitcoin markets. Quantitative Finance, 17, 703 - 716.

Liu, R., Tie, J., Wu, Z., & Zhang, Q. (2023). Pairs Trading: An Optimal Selling Rule with Constraints.

Mason, L., Baxter, J., Bartlett, P.L., & Frean, M. (1999). Boosting Algorithms as Gradient Descent. Neural Information Processing Systems.

Puspaningrum, H. (2012). Pairs trading using cointegration approach.

Sadorsky, P. (2021). A Random Forests Approach to Predicting Clean Energy Stock Prices. Journal of Risk and Financial Management.

Simon, N., Friedman, J.H., Hastie, T., & Tibshirani, R. (2013). A Sparse-Group Lasso. Journal of Computational and Graphical Statistics, 22, 231 - 245.

Ungever, C. (2015). Pairs Trading to the Commodities Futures Market Using Cointegration Method. Derivatives eJournal.

Tayal, C. (2021). Application of Machine Learning in Market- Neutral Pairs Trading in the Indian Stock Exchange.

Thazhungal Govindan Nair, S. (2021). Pairs trading in cryptocurrency market: A long-short story. Investment Management and Financial Innovations.

Wang, J. (2009). Pairs Trading with Robust Correlation.

Yan, T., Chiu, M.C., & Wong, H.Y. (2022). Pairs trading under delayed cointegration. Quantitative Finance, 22, 1627 - 1648.

Yu, Z. (2022). The Implementation of Support Vector Machine into Pairs Trading Strategy. Frontiers in Business, Economics and Management.

XGBoost (eXtreme Gradient Boosting). (2023). In Encyclopedia of Machine Learning and Data Mining. Springer. XGBoost. (n.d.). XGBoost Documentation. https://xgboost.readthedocs.io/en/latest/

Appendices

Appendix A: Turnitin Report

Copy of 53311_Chrispus Muhia_ Correlated Stock Identification in Pairs Trading Using XGBoost - DBMV11_Turnitin.docx.pdf

ORIGINALITY REPORT

13%	8%	8%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Strathmore University Student Paper	5%
2	www.mdpi.com Internet Source	<1%
3	www.researchgate.net Internet Source	<1%
4	deepnote.com Internet Source	<1%
5	fastercapital.com Internet Source	<1%
6	"Sustainable Development through Machine Learning, AI and IoT", Springer Science and Business Media LLC, 2023 Publication	<1%
7	drpress.org Internet Source	<1%

Appendix B: Ethical Review Clearance

Strathmore UNIVERSITY

25th March 2024

Mr Muhia Chrispus,
chrispus.njenga@strathmore.edu

Dear Mr Muhia,

RE: Correlated Stock Identification in Pairs Trading Using XGBoost

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** research proposal. Your application reference number is **SU-ISERC2081/24**. The approval period is from **25th March 2024 to 24th March 2025.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

STRATHMORE UNIVERSITY INSTITUTIONAL ACADEMIC AND ETHICAL REVIEW COMMITTEE
(SU-ISERC)
25-Mar-2024
Email:ethicsreview@strathmore.edu
P.O BOX 59857-00200
NAIROBI-KENYA