mathematics

Special Issue Reprint

New Advances in Computational Finance and Computational Intelligence in Finance

Edited by
Vasilios N. Katsikis, Xinwei Cao, Shuai Li, Dimitris Balios and Theodore E. Simos

mdpi.com/journal/mathematics

MDPI

New Advances in Computational Finance and Computational Intelligence in Finance

New Advances in Computational Finance and Computational Intelligence in Finance

Guest Editors

Vasilios N. Katsikis
Xinwei Cao
Shuai Li
Dimitris Balios
Theodore E. Simos

MDPI

Basel • Beijing • Wuhan • Barcelona • Belgrade • Novi Sad • Cluj • Manchester

Guest Editors

Vasilios N. Katsikis
Department of Economics
National and Kapodistrian University of Athens
Athens
Greece

Xinwei Cao
School of Business
Jiangnan University
Wuxi
China

Shuai Li
Faculty of Information Technology and Electrical Engineering
University of Oulu
Oulu
Finland

Dimitris Balios
Department of Economics
National and Kapodistrian University of Athens
Athens
Greece

Theodore E. Simos
Center for Applied Mathematics and Bioinformatics
Gulf University for Science and Technology
West Mishref
Kuwait

Editorial Office

MDPI AG
Grosspeteranlage 5
4052 Basel, Switzerland

This is a reprint of the Special Issue, published open access by the journal *Mathematics* (ISSN 2227-7390), freely accessible at: https://www.mdpi.com/journal/mathematics/special_issues/Computational_Finance_and_Computational_Intelligence_in_Finance.

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

Lastname, A.A.; Lastname, B.B. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range.

ISBN 978-3-7258-7462-0 (Hbk)
ISBN 978-3-7258-7463-7 (PDF)
https://doi.org/10.3390/books978-3-7258-7463-7

© 2026 by the authors. Articles in this reprint are Open Access and distributed under the Creative Commons Attribution (CC BY) license. The reprint as a whole is distributed by MDPI under the terms and conditions of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/).

Contents

Xingyao Song, Wenyu Chen and Junyi Lu
Research on Financial Stock Market Prediction Based on the Hidden Quantum Markov Model
Reprinted from: *Mathematics* **2025**, *13*, 2505, https://doi.org/10.3390/math13152505 . . . . . **1**

Xiaomei Feng and Song-Kyoo Kim
Statistical Data-Generative Machine Learning-Based Credit Card Fraud Detection Systems
Reprinted from: *Mathematics* **2025**, *13*, 2446, https://doi.org/10.3390/math13152446 . . . . . **18**

Guodong Hou, Dong Ling Tong, Soung Yue Liew and Peng Yin Choo
Comparative Analysis of Resampling Techniques for Class Imbalance in Financial Distress Prediction Using XGBoost
Reprinted from: *Mathematics* **2025**, *13*, 2186, https://doi.org/10.3390/math13132186 . . . . . **33**

Shiqing Qiu, Yang Wang, Zong Ke, Qinyan Shen, Zichao Li, Rong Zhang and Kaichen Ouyang
A Generative Adversarial Network-Based Investor Sentiment Indicator: Superior Predictability for the Stock Market
Reprinted from: *Mathematics* **2025**, *13*, 1476, https://doi.org/10.3390/math13091476 . . . . . **54**

Nadezhda Yarushkina, Aleksey Filippov and Anton Romanov
Contextual Analysis of Financial Time Series
Reprinted from: *Mathematics* **2025**, *13*, 57, https://doi.org/10.3390/math13010057 . . . . . **85**

Dimitris Lagios, Spyridon D. Mourtas, Panagiotis Zervas and Giannis Tzimas
A Weights Direct Determination Neural Network for International Standard Classification of Occupations
Reprinted from: *Mathematics* **2023**, *11*, 7, https://doi.org/10.3390/math11030629 . . . . . **102**

Vladislav N. Kovalnogov, Ruslan V. Fedorov, Dmitry A. Generalov, Andrey V. Chukalin, Vasilios N. Katsikis, Spyridon D. Mourtas and Theodore E. Simos
Portfolio Insurance through Error-Correction Neural Networks
Reprinted from: *Mathematics* **2022**, *10*, 3335, https://doi.org/10.3390/math10183335 . . . . . **116**

Chrysostomos Kasimis and Spyridon D. Mourtas
Exploiting Mean-Variance Portfolio Optimization Problems through Zeroing Neural Networks
Reprinted from: *Mathematics* **2022**, *10*, 3079, https://doi.org/10.3390/math10173079 . . . . . **130**

Bolin Liao, Zhendai Huang, Xinwei Cao and Jianfeng Li
Adopting Nonlinear Activated Beetle Antennae Search Algorithm for Fraud Detection of Public Trading Companies: A Computational Finance Approach
Reprinted from: *Mathematics* **2022**, *10*, 2160, https://doi.org/10.3390/math10132160 . . . . . **150**

mathematics

MDPI

Article

Research on Financial Stock Market Prediction Based on the Hidden Quantum Markov Model

Xingyao Song [1,*], Wenyu Chen [2] and Junyi Lu [3]

[1] Business School, The University of Sydney, Sydney 2000, Australia
[2] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[3] School of Physics, University of Electronic Science and Technology of China, Chengdu 611731, China
* Correspondence: songdd@uestc.edu.cn

Abstract

Quantum finance, as a key application scenario of quantum computing, showcases multiple significant advantages of quantum machine learning over traditional machine learning methods. This paper first aims to overcome the limitations of the hidden quantum Markov model (HQMM) in handling continuous data and proposes an innovative method to convert continuous data into discrete-time sequence data. Second, a hybrid quantum computing model is developed to forecast stock market trends. The model was used to predict 15 stock indices from the Shanghai and Shenzhen Stock Exchanges between June 2018 and June 2021. Experimental results demonstrate that the proposed quantum model outperforms classical algorithmic models in handling higher complexity, achieving improved efficiency, reduced computation time, and superior predictive performance. This validation of quantum advantage in financial forecasting enables the practical deployment of quantum-inspired prediction models by investors and institutions in trading environments. This quantum-enhanced model empowers investors to predict market regimes (bullish/bearish/range-bound) using real-time data, enabling dynamic portfolio adjustments, optimized risk controls, and data-driven allocation shifts.

Keywords: hidden Markov model (HMM); quantum machine learning; HQMM; stock forecasting

MSC: 81P68; 91G70

1. Introduction

With the rapid advancement of econometric finance, the financial industry faces significant challenges due to the exponential growth of data and the increasing complexity of financial models. To address these issues, numerous mathematical models have been developed and applied in the financial sector. While numerous mathematical approaches address these issues, their computational demands and inherent limitations underscore the need for innovative solutions.

For instance, reference [1] offers a framework for families to calculate their asset portfolios, serving as a valuable guide for many family asset managers. However, as the volume of data continues to increase, these methods have become somewhat overshadowed in recent years. In response to these challenges, artificial intelligence (AI) and machine learning (ML) [2,3] have emerged as powerful tools for navigating the demands of the modern data-driven era. Extensive research has been conducted in this domain. For

example, Ayyildiz and Iskenderoglu [4] introduced ML algorithms to predict movement trends of stock market indices in developed economies. While traditional machine learning methods demonstrate promise in stock index prediction, their scalability is constrained by exponentially growing data volumes. Consequently, quantum computing has emerged as a transformative solution to overcome these limitations by leveraging its computational efficiency advantages.

This paper introduces a novel stock prediction framework that employs a GMM-MVTF (Gaussian mixture models and means and variances to feature) discretization pipeline to overcome the inherent limitations of Gaussian mixture models (GMM) in handling continuous data for stock market predictions. By using the Shanghai Stock Exchange (SSE) and Shenzhen Stock Exchange (SZSE) datasets, this paper evaluates three HQMM variants: one with MVTF-discretized data and another with evenly discretized data, with performance comparisons against classical HMM under distinct processing techniques. This study unlocks a new dimension of quantum-enhanced computing for the industry and provides a valuable reference for further investigating the potential applications of quantum algorithms in other fields.

2. Related Work

2.1. Classical Machine Learning in Finance

Early ML applications focused on enhancing financial decision-making. Mahmoodi et al. [5] focused on using these algorithms to accurately forecast stock trading signals in financial markets. These studies highlight the immense potential of AI and ML in the financial sector, particularly in enhancing decision-making processes and improving portfolio management performance. Investing in stocks is a widely used strategy for portfolio diversification. However, it requires consideration of numerous factors, and individuals often encounter huge challenges when selecting stocks outside their areas of expertise. Identifying the most relevant factors for different stocks and addressing the biases introduced by specialized professional domains are necessary. Previous studies have made notable contributions to these issues. For example, Stambaugh [6] proposed a regression approach to investigate the relationship between lagged return rates and random regression quantities, which provides a viable framework for predicting and reporting various investment strategies and is able to mitigate regression interference brought by innovative regression quantities. Endri et.al. [7] aimed to develop an early warning system (EWS) model for predicting the delisting risk exposure of Islamic equities (ISSl) using a Support Vector Machine (SVM) with a set of financial variables. By optimizing SVM hyperparameters and employing hybrid architectures (e.g., SVM-deep learning), the model enhances predictive robustness, demonstrating the predictive power of financial indicators for delisting risk. Although AI and ML have proven to be valuable tools in stock market applications, researchers have observed that the computational power consumption of traditional ML methods has increased with the growing volume of market data [8]. Therefore, it is essential to explore other effective approaches to address the escalating consumption of data and computational resources.

2.2. Quantum Computing Approaches

Quantum methods demonstrate theoretical superiority in financial modeling. Due to its unique advantages of novel and efficient technology, quantum computing has emerged as an updated alternative to tackle existing challenges [9–11]. Jeswal and Chakraverty [12] stated that quantum neural networks are superior to classical methods across multiple domains such as SORNETTE, which explores a variety of models constructed based on the Ising model and extends them to elucidate numerous stylized features in financial markets,

and Bagheri et al. utilized the Quantum-behaved Particle Swarm Optimization (QPSO) algorithm to optimize the membership functions of the Adaptive Network-based Fuzzy Inference System (ANFIS) for forecasting financial time series, particularly in the foreign exchange market [13,14]. As a fundamental statistical method in time-series analysis, the hidden Markov model (HMM) is now playing a pivotal role in the stock market by considering its various hidden characteristics. Early in 2011, Guidolin [15] comprehensively reviewed the applications of the Markov transformation models in empirical finance, such as the analyses of stock returns, the term structure of default-free interest rates, the dynamics of exchange rates, and the joint process of stock and bond returns. Based on the quantum conditional master equation, a new quantum ML algorithm, split hidden quantum Markov model (SHQMM), was proposed by Li et al. [16,17], which further expands the potential applications of quantum computing in complex systems.

While HQMM [18] theoretically addresses these gaps, no prior work has implemented them for stock prediction due to the unsolved challenge of continuous data discretization. Against the backdrop of growing research interest in quantum machine learning for financial time-series problems, this paper proposes a quantum-based prediction algorithm for the stock market, built upon the HQMM. Leveraging the unique characteristics of quantum computation, the proposed HQMM-based algorithm is compared with the classical HMM to highlight the advantages of quantum coherence. The study involves data collection and processing and addresses the limitations of handling continuous data through Gaussian mixture models (GMM) and means and variances to feature (MVTF) analysis. Model training likelihood function is constructed based on temporal features. The model is evaluated and optimized using the distance and accuracy (DA) algorithm, thereby enabling the systematic demonstration of AI quantitative algorithms' applied value in accelerating model training, autotuning hyperparameters, enhancing feature retention, dynamically adapting to trading environments, and generating investment-actionable insights.

3. The Comparison Between the Classical HMM and HQMM

3.1. Hidden Markov Model in Stock Forecasting

HMM describes a hidden Markov chain that generates an unobservable sequence of states, where the state probability at time t, $P(y_t)$, is only determined by the previous state y_{t-1}, as (1) shows:

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_t) = p(y_t|y_{t-1}) \tag{1}$$

By using the time-independent constant matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ to define the transition matrix and the observation matrix, respectively, in the Markov process, the model can be defined as $\lambda = (\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\pi})$ [19], which denotes the initial state distribution $\boldsymbol{\pi}$ [20]:

$$\boldsymbol{A} = [a_{i,j}]_{N\times N} \tag{2}$$

$$\boldsymbol{C} = [c_{i,j}]_{N\times M} \tag{3}$$

$$\boldsymbol{\pi} = [\pi_i]_N \tag{4}$$

where there are the elements at the i-th row and the j-th column of the $N \times N$ transition matrix and the $N \times M$ observation matrix, respectively. The initial state distribution π_i defines the probability of being in state i at the first time point. When the HMM is applied to stock market prediction, elements such as the risk-free interest rate, systemic risk, and risk premium are incorporated into the hidden state, and the observation matrix represents the price of each stock. By using historical daily stock prices as input data, the transition matrix $\boldsymbol{A}$ and observation matrix $\boldsymbol{C}$ of HMM are trained by using ML techniques. Then, the probability of each subsequent hidden state can be derived from matrix $\boldsymbol{A}$, while the

probability of the most possible stock price at the next time step [19,21] can be calculated using the following equation:

$$\delta_{t+1}(i) = \max_{1 \le j \le N}[\delta_t(j)a_{j,i}]c_i(o_{t+1}) \tag{5}$$

Given model $\lambda = (\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\pi})$ and observation sequence = $\{o_1, o_2, \ldots o_T\}$, o_{t+1} is the observation vector o at time $t+1$, where $\delta_{t+1}(i)$ is the predicted state with the largest probability at time $t+1$. Obviously, this model only focuses on the price variations of a single stock over time but ignores other influencing factors. However, in the real stock market, numerous factors have a determinant influence on stock prices, including both external and internal variables, such as risk-free interest rate, systemic risk, and risk premium. This renders HMM-based predictions fundamentally unstable in real environments.

3.2. Hidden Quantum Markov Model

As an extension of the classical HMM framework, HQMM [22] can be expressed as follows:

$$p(y_t|\rho_{t-1}) = \mathrm{tr}(K_{y_{t-1}}\rho_{t-1}K_{y_{t-1}}^{\dagger}) \tag{6}$$

$$\rho_t = \frac{K_{y_{t-1}}\rho_{t-1}K_{y_{t-1}}^{\dagger}}{\mathrm{tr}(K_{y_{t-1}}\rho_{t-1}K_{y_{t-1}}^{\dagger})} \tag{7}$$

where y_t is the observation state, ρ_{t-1} is the known state, $p(y_t|\rho_{t-1})$ is the conditional probability of the observation state at time t when the density matrix ρ at time $t-1$ is known, tr is the trace, $K_{y_{t-1}}$ is the Kraus operator corresponding to the observation state at time $t-1$, and the superscript † indicates the conjugate.

HQMM is trained using gradient descent or variational inference to optimize quantum operators by density matrix ρ (showing the hidden state) and state update (controlled by the Kraus operator $\{Km\}$, following quantum channel constraints, where m is the number of the operator, and I refers to the identity matrix). Through the global information storage of quantum states, interference effects, long-term dependency modeling, and exponentially scalable state space, HQMM breaks through the limitations of HMM, which can only focus on a single stock trend, making it more suitable for multi-asset, market linkage analysis and intricate financial systems.

Both HMM and HQMM are capable of inferring the probability of the next state from the previous state. In a classical HMM, the next hidden state and observation state can be inferred from the transition matrix $\boldsymbol{A}$ and the state observation matrix $\boldsymbol{C}$, respectively. Comparatively, in HQMM, the Kraus operator replaces the matrices $\boldsymbol{A}$ and $\boldsymbol{C}$, serving to predict the next state. Traditional HMMs can only model Markov processes, where the current state depends solely on the previous state, and a fixed probability transition matrix governs transitions. In contrast, Kraus operators in HQMMs leverage density matrices to retain historical information and quantum coherence. Additionally, HQMMs allow the use of multiple Kraus operators, each of which can correspond to different market conditions or trading patterns, providing greater flexibility in capturing complex financial dynamics. A comparison of the two models is presented in Table 1, and the relevant formulas and characteristics can be found in Ref. [22].

In classical HMM probability, the vector $\vec{x}$ represents the probability of different market states and is referred to as a probability vector. The belief state reflects the confidence level in the market being in a specific state at a given moment—such as rising, stable, or falling. The joint distribution $\vec{x}_1 \otimes \vec{x}_2$ encodes the combined probability of two or more variables occurring simultaneously. Marginalization $\vec{x} = \sum_d \vec{x} \otimes \vec{d}$ refers to the process

of ignoring a variable d in order to focus only on the distribution over x. Conditional probability $P(\vec{x}|d) = \frac{P(d,\vec{x})}{P(d)}$ is used to update belief about x, given new information d.

Table 1. The comparison between the classical HMM and HQMM [22].

HMM Probability		HQMM Analogue	
Description	**Representation**	**Description**	**Representation**
Belief State	$\vec{x}$	Density Matrix	$\hat{\rho}$
Joint Distribution	$\vec{x}_1 \otimes \vec{x}_2$	Multi-particle Density Matrix	$\hat{\rho}X_1 \otimes \hat{\rho}X_2$
Marginalization	$\vec{x} = \sum_d \vec{x} \otimes \vec{d}$	Partial Trace	$\hat{\rho}X = tr_D(\hat{\rho}XD)$
Conditional probability	$P(\vec{x}\|d) = \frac{P(d,\vec{x})}{P(d)}$	Projection + Partial Trace	$P(state\|d) = tr_D(\hat{P}_d\hat{\rho}XD\hat{P}_d^\dagger)$

In the HQMM analogue, $\hat{\rho}$ represents a density matrix, which incorporates not only probabilistic information but also quantum coherence. Density matrix allows for interference between different states, making it more suitable for capturing the complexity of financial markets. A multi-particle density matrix, $\hat{\rho}X_1 \otimes \hat{\rho}X_2$, models the joint behavior of strongly correlated variables—such as two closely linked sectors—more accurately. A partial trace, $\hat{\rho}X = tr_D(\hat{\rho}XD)$, is used when macro-level information (D) is known and focuses on a particular asset (X), obtaining partial trace for the macro-variable (D). Projection+partial trace constitutes the quantum analogue of conditional probability: the system is first projected onto a measured outcome d (e.g., a major policy announcement), and then a partial trace is taken to infer its effect on the market state X (e.g., stock movement).

Fundamentally, HMM and HQMM differ in their modeling approach. HMM forecasts market behavior by learning transition probabilities between historical price states. In contrast, HQMM leverages the principles of quantum mechanics—superposition (interference between rising and falling probabilities) and entanglement (interlinked behavior of assets like U.S. and Chinese stocks)—to potentially capture more intricate market dynamics. This includes nonlinear dependencies, long-range memory effects, and hidden correlations that classical models may fail to represent.

4. Experimental Procedure Introduction

To achieve accurate prediction of the stock market based on HQMM, three main steps should be performed, involving the collection of real-time stock data, conversion of the continuous data into discretized formats, and training of HQMM based on the processed data to predict stock trends, as illustrated in Figure 1.

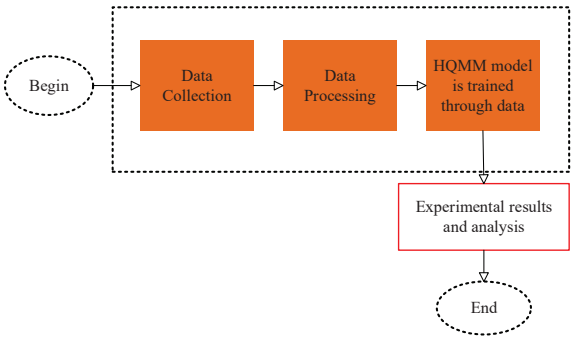


Figure 1. The framework of stock forecasting by quantum hidden Markov.

4.1. Data Collection

Here, 15 stocks with stock codes of 000901, 000036, 000031, 601318, 600519, 600009, 600004, 600050, 600022, 000623, 000401, 000488, 000518, 000800, and 000895 were taken into consideration. The 5 key indices of these stocks, opening price, highest price, lowest price, closing price, and trading volume, were collected daily from the real-time stock data of SSE and SZSE from June 2018 to June 2021. Financial data were accessed via the Tushare Application Programming Interface (API). The workflow integrates multiple specialized libraries: backtrader and Tushare for stock data acquisition, pandas and NumPy for data preprocessing, hmmlearn and scikit-learn for classical modeling (hidden Markov models and Gaussian methods), and Qiskit for quantum circuit implementation. The following data were split into different time periods: 1 June 2018–26 October 2019 (training period), 30 October 2019–24 August 2020 (validation period), and 24 August 2020–23 June 2021 (testing period).

Each stock has an index number of 5. These indices serve as the evaluation standards for investment performance and the innovation of index derivative products, providing the foundational conditions for index investment.

4.2. Data Processing

As discussed before, the current HQMM-based algorithm can only handle discretized time-series data [21]; thus, the obtained continuous data from SSE and SZSE cannot be directly used for this hidden quantum Markov prediction. To ensure that the data can be compatible with HQMM, further data processing is required. As for HMMs, when continuous data is applied, integrated algorithms, such as Gaussian HMM, are usually employed. However, an equivalent quantum version of such an approach is not accessible for HQMM, necessitating the exploration of alternative methods. Discretizing continuous data is often considered the first option for HQMM; however, it also faces a significant challenge. The original data may be seriously distorted, having a negative impact on the model's effectiveness. As a result, another method—integrating the Gaussian mixture model (GMM) with the means and variances to feature (MVTF) model—is proposed and also applied for data processing, aiming at ensuring maximum fidelity in financial data representation and thus enhancing the model performance.

4.2.1. Even Discretization of Continuous Data

- In order to better discretize continuous data, the traditional discretization method in Table 2 is presented, and then the experimental method in Table 3 is improved to reduce external interference. Table 2 demonstrates that the data underwent straightforward processing and were subsequently input into the HQMM model. While the prediction results were generated, they were significantly influenced by external factors, leading to a lack of robustness and vulnerability to artificial interference. To enable a fair comparison with our improved method, the same approach was employed, whereby the obtained data were categorized into three intervals based on stock price fluctuations. These intervals were represented by the integer values 1, 3, 5, 2, 4, and 6, respectively (Table 3). The value of L is mentioned in formula 8 (through the calculation of GMM mean and variance and MVTF below), and L is converted into the value of S(Observation status in Table 3) and then transformed into a bar graph in Figure 2. The ordinate of the green bar chart is the value of separate S, the dotted line represents the real stock trend value, and the solid line is the line of the midpoint of the bar chart, indicating that the stock trend obtained through the division of S is more realistic.

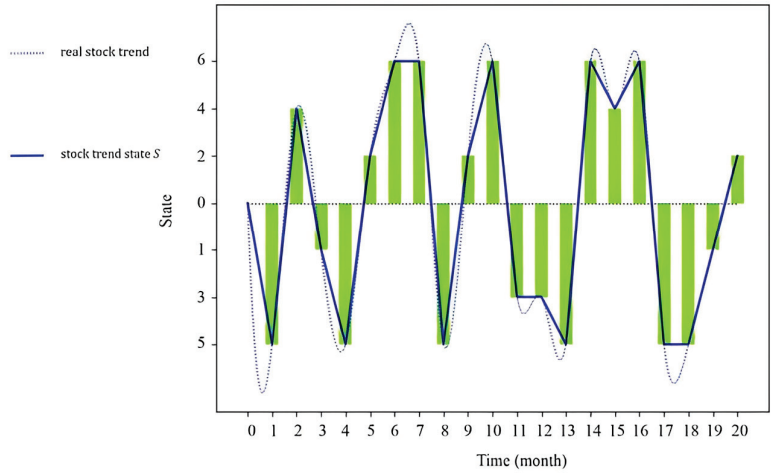


Figure 2. Stock trend state with code 600519 obtained by the MVTF model during the 20 months.

Table 2. Classify stock data into integers.

Growth Rate	Rise	Down
0~5	1	2
5~10	3	4
10~∞	5 (bull market)	6 (bear market)

Table 3. According to the classification of the value of L, it is converted to the value of S.

Range	Value of *S*	Range	Value of *S*
$-20 < L \leq 0$	1	$0 < L \leq 20$	2
$-50 < L \leq -20$	3	$20 < L \leq 50$	4
$L \leq -50$	5	$L \geq 50$	6

The calculation of stock price fluctuations (increase or decrease) is performed by subtracting the previous day's closing price from the next day's closing price. If the resulting value *S* is positive, it indicates a rise; conversely, a negative *S* signifies a fall. This approach facilitates the data input into the HQMM model and simplifies the subsequent analysis of experimental results.

4.2.2. Data Discretization with GMM and MVTF

The processing steps of our proposed method, which combines GMM with MVTF to discretize continuous data, are as follows:

(1) The data collected during a certain time period are first inputted into the GMM-based ML algorithm, allowing for the training of multiple Gaussian distributions. The common steps for GMM to process data are as follows: (i) The dataset and the variables to cluster are selected. (ii) The optimal cluster number *K* is determined based on the characteristics and size of the dataset using the Bayesian Information criterion (BIC) or Akaike Information criterion (AIC) through multiple experiments. The cluster parameter *K* is empirically determined as 6 in this experimental setup. (iii) Random initialization or k-means algorithm is used to initialize GMM parameters. (iv) The expectation maximization (EM) algorithm is used to assign each sample to the Gaussian distribution with the highest probability, and the probability of each sample belonging to the Gaussian distribution is calculated (posterior probability, E-step). (v) M-step in the EM algorithm is used to find the mean value, method, and weight. (vi) Steps iv and v are repeated until the posterior probability change is less than a certain value. (vii) The mean and variance of each distribution are extracted based on the resulting distribution.

- In an experimental model, the collected data is accurately quantified and decomposed into k models based on the Gaussian probability density function, as shown in Figure 3. (Schematic diagram illustrates GMM, so no scale is required. The red line is the overall trend, which then breaks down into the blue line). Here, a one-dimensional GMM is utilized, the 30-day data is considered and input into a Gaussian mixture model for training, and n is set to 5, where n represents the number of Gaussian mixture components. In order to ensure consistent generalization across all data samples and to avoid the imbalance phenomenon where some data perform exceptionally well while other data significantly deteriorate due to parameter deviations, the above parameters were selected. This means that the GMM model learns five Gaussian distributions and obtains variances $(\sigma_1, \sigma_2, \cdots, \sigma_5)$ and means $(\mu_1, \mu_2, \cdots, \mu_5)$ of these Gaussian distributions.

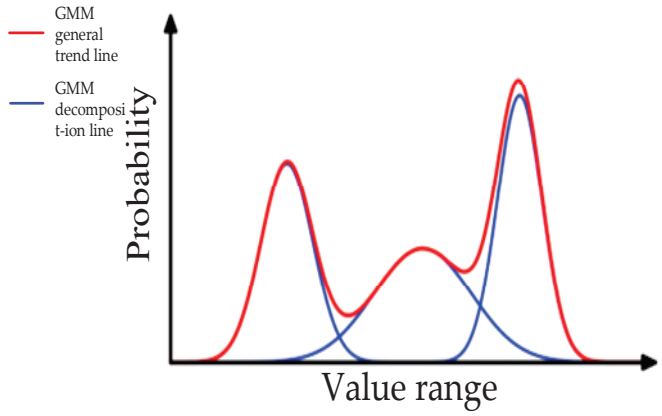


Figure 3. GMM diagram.

(2) The means and variances obtained in the previous step are further processed through the MVTF model and then used to determine the trend states of the stocks over the specified period. During this step, how to convert the before-derived means and variances into the stock trend states (indicating a bull or bear market) is crucial because it has a significant effect on the fidelity degree of the experimental data when input into HQMM. To maximize this fidelity, the following conversion processes are performed:

- The n variances $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ obtained from the experimental data are sorted from small to large and assigned to $a_1, a_2, \cdots, a_n$ in order, and the mean μ_i corresponding to σ_i is also assigned to $b_1, b_2, \cdots, b_n$ in the same order as σ.
- The L value of each set of data is calculated using the following equation:

$$\mathrm{L} = \sum_{i=1}^{n} b_i(i * 5\% + 5\%) \tag{8}$$

- b_i represents the mean μ_i corresponding to the i-th variance a_i. The trend state S of stocks can be defined by classifying the value of L (L is a calculated value based on b_i and a weighting coefficient), based on the classification principle shown in Table 3. To ensure MVTF fully leverages all information from the GMM-generated Gaussian distributions while accounting for the correlation between experimental results and the number of Gaussians, Formula (8) is derived by using a criterion. As a result, the stock trend state S is made into tabular data. Correspondingly, the resulting trend state of stock 600519 is shown in Figure 2.

Besides the above processes involving GMM and MVTF, the whole operation flow of data processing also includes: (i) collecting real-time stock data; (ii) calculating and

tabulating the difference between the closing price of each day and that of the previous day; (iii) splitting all collected data into a group of 30 days; (iv) inputting each group of data into GMM and setting the number of Gaussian distributions in GMM to n (n is an integer and $n \geq 1$); (v) calculating the means and variances of n Gaussian distributions; and (iv) converting these means and variances into the stock trend state through the MVTF model. After these operations, the collected continuous financial market data is successfully discretized and finally converted into a format accessible to HQMM.

4.3. HQMM Training

In order to accurately predict the stock market with the previous data processing results, the HQMM model is constructed. How to construct the likelihood function and solve the Kraus operator with the gradient descent method is at the core of this model. The historical stock observation data $S_1, S_2, \ldots, S_n$ are used to construct the likelihood function [23] as follows:

$$\varsigma = \ln \mathrm{tr}(K_{S_n} \cdots (K_{S_1} \boldsymbol{\rho}_0 K_{S_1}^{\dagger}) \cdots K_{S_n}^{\dagger}) \tag{9}$$

where $\boldsymbol{\rho}_0$ is the manually set initial density matrix. In the initial state $\boldsymbol{\rho}_0$ under different K values, it exhibits the advantage of robustness, allowing the model state to evolve from any initial state to a stable state. Additionally, the manually set method for HQMM is the same used in References [16,23]. During the HQMM process, when the Markov chain has a long sequence, $\boldsymbol{\rho}_0$ will produce a decoherence effect. In the process of evolving the relative state, for the first half of a batch, any initial state undergoes evolution, while the second half undergoes the experiment. In this experiment, $\boldsymbol{\rho}_0$ can be set to the following: $\begin{pmatrix} 8.0360857 \times 10^{-2} & -3.4289963 \times 10^{-2} \\ -3.4289963 \times 10^{-2} & 9.1963914 \times 10^{-1} \end{pmatrix}$.

Then, the likelihood function (9) is used to calculate the derivative of all possible Kraus operators for gradient descent to maximize the value of the likelihood function. Therefore, a matrix solution to the Kraus operator K_{S_n} is obtained, which is the n-th Kraus operation matrix. Each observation value corresponds to an operation matrix, which is the same at the beginning; the solution of the derived HQMM is transformed into a constrained optimization problem: relational matrix ς is the maximum and $\sum K_m^{\dagger} K_m = I$ [24], with m being the number of the operator and I referring to the identity matrix. Then, the likelihood function is used to conduct derivative processing on all the Kraus algorithms. Subsequently, the gradient descent method is employed to calculate the maximum likelihood function value, thereby enabling the determination of the matrix solution of the first Kraus.

For the convenience of computer operation, a new matrix $\boldsymbol{Z}$ [24] is reconstructed by stacking all the 2×2 Kraus operator matrices, as shown in Figure 4. It has a dimension of $2H \times 2$, determined by the output number H, i.e., the number of stock observation states. The constrained optimization problem requires the constructed matrix $\boldsymbol{Z}$ to satisfy the following: ς is the maximum and $\boldsymbol{Z}_m^{\dagger} \boldsymbol{Z}_m = I$ [24]. Since $\boldsymbol{Z}$ is on the Stiefel manifold, its constrained optimization problem can be transformed into the following unconstrained problem, which can be solved by the gradient descent algorithm: $G = \partial \varsigma / \partial Z$ [24] and $Z = Z - \tau \boldsymbol{U}\left[I + (\tau/2)\boldsymbol{V}^{\dagger}\boldsymbol{U}\right]^{-1} \boldsymbol{V}^{\dagger} Z$ [24]. Here, G is the partial differential of the likelihood function for $\boldsymbol{Z}$; τ is the step size of gradient descent (a real number in the interval [0, 1]); and both $\boldsymbol{U}$ and $\boldsymbol{V}^{\dagger}$ are the augmented matrices and represented by $[G|Z]$ and $[Z|-G]$ [24]. By solving this constrained optimization problem, the Kraus operator can be solved, the latest $\boldsymbol{\rho}_t$ is updated, and the whole HQMM is obtained. Finally, the result of reversing $\boldsymbol{Z}$ to obtain the Kraus operators is derived, and the quantum hidden Markov model is used to solve and obtain the prediction of the stock's trend, $P(S_t|\rho_{t+1})$, at the next moment, based on the results of the Kraus operators.

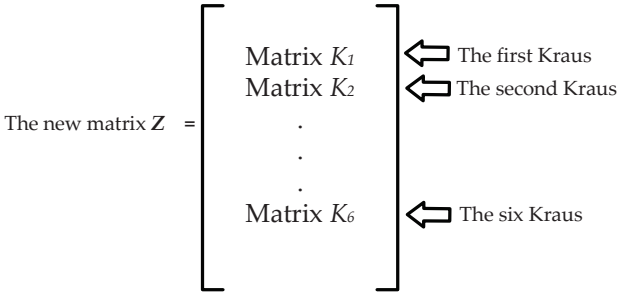


Figure 4. Schematic diagram of constructing a new matrix **Z**.

To accurately evaluate the model's performance, the description accuracy (DA) must be obtained. The fundamental principle of DA is to serve as a standardized log-likelihood ratio metric. It achieves likelihood standardization with respect to state-space dimension and length normalization, enabling fair comparison across models of varying complexity. For more detailed explanations, refer to reference [22]. The algorithm is adopted and defined as follows:

$$\mathrm{DA} = f\left(1 + \frac{\log_w P(Y|D)}{r}\right) \tag{10}$$

where r is the length of the sequence, w is the number of hidden states, Y is the observation data, D is the generated model, and $f(\cdot)$ is a nonlinear function that takes the argument from $(-\infty, 1]$ to $(-1, 1]$:

$$f(x) = \begin{cases} x & , x \geq 0 \\ \frac{1-e^{-0.25x}}{1+e^{-0.25x}} & , x < 0 \end{cases} \tag{11}$$

If DA = 1, the model perfectly predicts the stochastic sequence, while $1 > \mathrm{DA} \geq 0$ means that the model predicts the sequence better than random [22]. If DA < 0, the results indicate that the model performs poorly and is inferior to a random baseline. The more DA converges to 1, the better the model functions. It describes the DA as the loss function used to evaluate the quality of the model training. In repeated experiments, DA demonstrates noise resistance in measurement, with its values eventually converging to a stable state [16]. Subsequently, through the model, the influence of different initialization positions of Kraus (equivalent to data noise) on the training results of the model is calculated, as shown in Figure 5 [16]. The RS represents the random number seed.

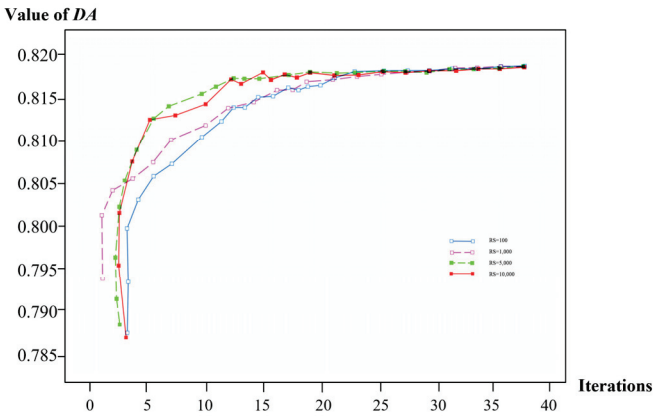


Figure 5. The influence of different initialization positions of Kraus on the training results of the model.

5. Experimental Results and Discussion

Since different stocks and lengths of prediction time would lead to different prediction performances for both HQMM and HMM, 15 distinct stocks were selected for analysis, and

the average performance was calculated to evaluate the models' performance. During the experiment, for the HQMM model constructed in this paper, the number of hidden states was 6, which was derived through global optimization, while the settings for the Kraus operators and density matrices were retained from prior configurations. This methodology ensures robustness and stability across the full dataset, thereby eliminating performance fluctuations induced by data distribution discrepancies. Additionally, the matrix $\boldsymbol{Z}$ with a dimension of 12 $\times$ 2 is obtained from the training process. The calculated results of DA for the evenly-discretized continuous data and that processed with MVTF are shown in Figure 6a,b, respectively. It can be found that when the number of iterations is less than 5, the value of DA increases rapidly, while once the number of iterations exceeds 40, the value of DA tends to converge. At this time, the mode training is completed.

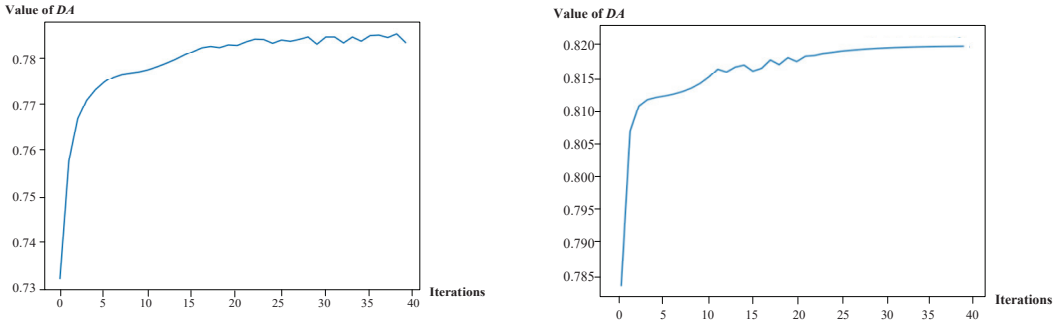


Figure 6. DA results obtained by (**a**) evenly discretizing processing data and (**b**) GMM and MVTF processing data.

Upon completing model training, the following steps are further conducted to investigate the model's prediction performance:

- Based on the constructed matrix $\boldsymbol{Z}$, six matrices $\boldsymbol{K}_1, \boldsymbol{K}_2, \cdots, \boldsymbol{K}_6$ are obtained, and the corresponding prediction values $J_1, J_2, \cdots, J_6$ could be calculated by the following equation:

$$J_n = -\ln\left[\mathrm{tr}\left(\boldsymbol{K}_n \boldsymbol{\rho}_t \boldsymbol{K}_n^{\dagger}\right)\right] \tag{12}$$

- The minimum value J_n' is found among $J_1, J_2, \cdots, J_6$, and thus the subscript n ($n \in [1, 6]$) is the predicted state at the t time. For example, if the minimum value is J_3', 3 is the most possible next state (E).
- By setting the initial capital (IC) to 1,000,000, the trend of each stock is predicted one by one with different methods. If the prediction result of one stock is rising, i.e., J_n' is J_2', J_4', or J_6', buy(long) it; otherwise, sell(short) it. if the prediction result is falling (J_n' is J_1' J_3', or J_5'). In order to measure the prediction effect, for the following scheme, the final capital (FC) that the operator (including buying and selling at the right time) uses for HQMM can be obtained. It should be noted that the transaction costs in the whole process are neglected. The core function of HQMM is to predict market state evolution by forecasting the most probable range of price fluctuations, and it is decoupled from trade execution. If HQMM needs to be integrated into a trading system for downstream applications, this can be achieved in the future through a portfolio optimization module.

To compare the prediction result of the stock with the natural growth capital (NC), the comparative growth rate (R) is proposed [25], formulated as follows:

$$R = \frac{\mathrm{FC} - \mathrm{NC}}{\mathrm{FC}} \times 100\% \tag{13}$$

$$NC = IC \times (FP/IP) \tag{14}$$

$$FC = FC \times (SP/BP) \tag{15}$$

where IP is the initial price of each stock, FP is the final price of each stock at the time of experiment, SP is the selling price, and BP is the buying price. Obviously, NC can be obtained from the real data of the stock, as shown in Table 4.

Table 4. Real trend of the stocks in a certain period.

Stock Code	Period (Day)	IC (CNY)	NC (CNY)	Change Rate
000901	90	1,000,000	1,716,216	71.62%
000036	90	1,000,000	1,359,855	35.98%
000031	90	1,000,000	1,384,458	38.44%
601318	90	1,000,000	1,391,863	39.18%
600519	90	1,000,000	1,409,882	40.98%
600009	180	1,000,000	1,290,323	29.03%
600004	180	1,000,000	1,516,869	51.68%
600050	180	1,000,000	879,710	−12.03%
600022	180	1,000,000	774,869	−22.52%
000623	180	1,000,000	8,495,625	−15.05%
000401	360	1,000,000	1,032,701	3.27%
000488	360	1,000,000	1,406,417	40.64%
000518	360	1,000,000	1,527,331	52.73%
000800	360	1,000,000	1,506,112	50.61%
000895	360	1,000,000	2,104,429	110.44%

- At the beginning, the final capital (FC) is set equal to the initial capital (IC), and then the selling price (SP) and buying price (BP) are defined accordingly. Once the HQMM indicates a buying suggestion, the calculation of (15) is repeatedly performed, leading to the update of the final capital (FC).

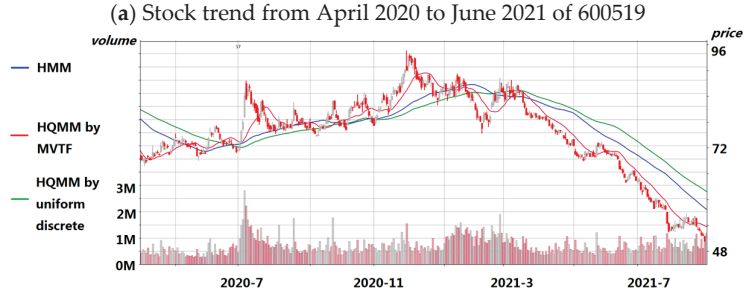
Regarding the stock prediction using HMM, the detailed processes can be found in reference [26]. This involves using classical ML techniques and historical stock data as input to comprehensively train the HMM model, which is then utilized to predict stock trends. Similarly, both FC and R can also be obtained with the same methods in HQMM.

To further investigate the model's time generalization capabilities, three different prediction periods of 90 days, 180 days, and 360 days were considered to study the impacts of different stocks, time periods, and models on prediction results.

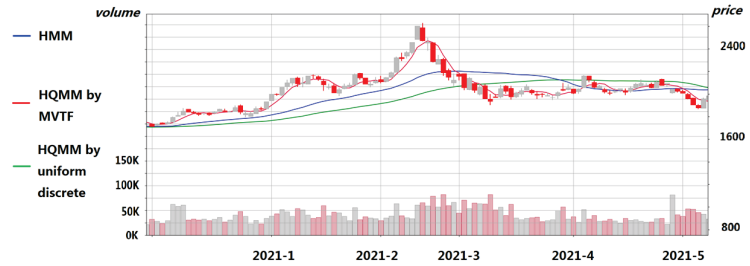
Three randomly selected stocks were taken as an example, and the predicted trends by three different methods were compared with the actual trends over different time periods. The results are shown in Figure 7 (The grey columns represent an increase, while the red columns represent a decrease). For the stock with code 600519, the predicted trend from August 2020 to July 2021, as determined by HMM, best aligns with the actual trend, as shown in Figure 7a. For the stocks with codes 601318 and 600022, HQMM performs the best, as shown in Figure 7b,c. Notably, for all these stocks, the prediction results of HQMM with the data discretized by GMM and MVTF are always superior to those of HQMM with evenly discretized data. This further demonstrates that selecting multiple stocks and various time periods to evaluate the model's average prediction capability is necessary. By applying similar operations to our selected 15 stocks and considering different time periods, the (FC) and (R) are calculated by three different prediction methods, and the detailed results are summarized in Tables 4–7.



(**a**) Stock trend from April 2020 to June 2021 of 600519



(**b**) Stock trend from June 2020 to June 2021 of 601318



(**c**) Stock trend from January to April 2021 of 600022

Figure 7. Comparison of the stock trends from three different prediction methods with the real trend.

Table 5. Prediction results of HMM.

Stock Code	Period (Day)	IC (CNY)	NC (CNY)	R (%)
000901	90	1,000,000	1,802,092	5.00
000036	90	1,000,000	1,370,892	0.81
000031	90	1,000,000	1,393,821	0.67
601318	90	1,000,000	1,443,975	3.74
600519	90	1,000,000	1,422,745	0.91
600009	180	1,000,000	1,377,119	6.72
600004	180	1,000,000	1,539,977	1.52
600050	180	1,000,000	929,570	5.66
600022	180	1,000,000	781,347	0.83
000623	180	1,000,000	904,480	6.46
000401	360	1,000,000	1,109,338	7.42
000488	360	1,000,000	1,492,568	6.12
000518	360	1,000,000	1,603,722	5.00
000800	360	1,000,000	1,572,856	4.43
000895	360	1,000,000	2,118,394	0.66

Table 4 presents the (NC) and its variation rate for 15 stocks across different periods (90, 180, and 360 days), serving as the benchmark for evaluating prediction model performance. A model demonstrates practical value if its forecast-guided (FC) exceeds (NC). Table 5 displays the predictive performance of the classical HMM, with the (R) as the key metric. Table 6 presents the prediction results of HQMM with evenly discretized data, also evaluated using (R). Table 7 presents HQMM's performance when employing GMM and MVTF transformation, with R as the comparative metric. (With an investment

of 1,000,000 in 90 days, the natural capital of stock code 000901 is 1732049. The growth rate of this model, compared with that of natural growth capital, is 0.92%.)

Table 6. Prediction results of HQMM with evenly discretized data.

Stock Code	Period (Day)	IC (CNY)	NC (CNY)	R (%)
000901	90	1,000,000	1,732,049	0.92
000036	90	1,000,000	1,366,493	0.48
000031	90	1,000,000	1,388,911	0.32
601318	90	1,000,000	1,422,762	2.21
600519	90	1,000,000	1,416,535	0.47
600009	180	1,000,000	1,311,691	1.65
600004	180	1000,000	1,530,753	0.91
600050	180	1,000,000	890,753	1.25
600022	180	1,000,000	776,823	0.25
000623	180	1,000,000	868,408	2.21
000401	360	1,000,000	1,101,913	6.70
000488	360	1,000,000	1,484,485	5.55
000518	360	1,000,000	1,537,752	0.68
000800	360	1,000,000	1,513,937	0.51
000895	360	1,000,000	2,100,561	−0.18

Table 7. Prediction results of HQMM with the discretized data by MVTF.

Stock Code	Period (day)	IC (CNY)	NC (CNY)	R (%)
000901	90	1,000,000	1,794,014	4.53
000036	90	1,000,000	1,373,676	1.01
000031	90	1,000,000	1,410,884	1.90
601318	90	1,000,000	1,431,247	2.82
600519	90	1,000,000	1,444,693	2.46
600009	180	1,000,000	1,368,929	6.09
600004	180	1,000,000	1,551,502	2.52
600050	180	1,000,000	915,441	4.06
600022	180	1,000,000	797,216	2.88
000623	180	1000,000	928,868	9.33
000401	360	1,000,000	1,199,080	16.11
000488	360	1,000,000	1,534,565	9.11
000518	360	1,000,000	1,639,544	7.34
000800	360	1,000,000	1,561,505	3.67
000895	360	1,000,000	2,117,593	0.62

(NC) refers to the value attained by initial capital under natural market conditions without model-guided investment, reflecting intrinsic stock value growth and providing a performance baseline. (R) quantifies the difference between model-predicted returns and (NC), calculated as $R = \frac{\text{FC}-\text{NC}}{\text{FC}} \times 100\%$, where (FC) represents final capital. This metric measures the growth rate of return generated by model-driven trading strategies relative to passive market growth.

Experimental validation demonstrates the significant advantages of the quantum-enhanced model over traditional approaches. Table 8 compares the prediction results and performances of the above-mentioned three prediction methods. (1) Superior predictive performance (average capital with shorting/longing as predicted): 1390860 is realized by classical HMM, 1362922 by HQMM with evenly discretized data, and 1404827 by HQMM with data discretized by MVTF. (2) Average comparative growth rates: 3.73%, 1.59%, and 4.96% (Formula (12)), respectively. (3) Total required time (min): 40, 12, and 12. (4) The computational complexity: HMM = $O(Tn^2)$, HQMM = $O(Tn\text{w})$ [16]. (T denotes the

prediction sequence length, n represents the number of hidden states, and w corresponds to the observable space dimension.) This linear scaling in w versus HMMs' quadratic dependence on n demonstrates HQMMs' computational advantage for high-dimensional observation spaces.

Table 8. Result and performance comparisons of three different prediction methods.

Prediction Method	Average Capital Under No Operation (CNY)	Average Capital with Shorting/Longing as Predicted (CNY)	Average R (%)	Number of Hidden States	Total Time Required (min)
HMM	1,343,373	1,390,860	3.73	2–6	40
HQMM with evenly discretized data	1,343,373	1,362,922	1.59	2	12
HQMM with discretized data by MVTF	1,343,373	1,404,827	4.96	2	12

The MVTF-discretized HQMM approach demonstrates superior performance over comparative methods, with 70% faster computation (12 vs. 40 min), a 4.96% higher comparative growth rate, and a computational advantage. By employing a novel GMM-MVTF framework that converts continuous financial data into discrete time-series sequences, this methodology effectively resolves inherent processing limitations in hidden quantum Markov models.

Practical value discussion: Due to the clear advantages demonstrated by the MVTF-discretized HQMM experiments, it is now necessary to discuss its practical significance. (1) Real-time signal generation identifies bullish, bearish, or range-bound regimes to support pre-trade decisions through predictive alerts. (2) Enhanced quantitative strategies strengthen in-process risk management via automatic stop-loss triggers and maximum drawdown control when hidden state probabilities indicate market deterioration, enabling optimized portfolio hedging. (3) Dynamic optimization of asset allocation thresholds via asset rotation gradually increases exposure to growth stocks during predicted bull markets (shifting to high-dividend companies during bear markets), enabling optimized portfolio hedging. (4) It is supplemented by sector rotation forecasts from HQMM that refine industry heatmaps. (5) Quantum-driven model accelerates processing speeds (70% faster), significantly augmenting human analytical capabilities. (6) Practical implementability is ensured through MVTF-discretized quantum algorithms trained on real-world financial time series. (7) In the future, it can also be integrated with the development of other assessment frameworks in a collaborative manner.

6. Conclusions

To address the limitations of HQMM, which does not apply to continuous data, this paper proposes a data discretization method for stock data while also applying the HQMM approach to forecast stock trends for the first time. A series of experiments were conducted based on the real data collected from SSE and SZSE to evaluate the performance of such quantum-inspired classical algebraic models. Experimental results demonstrate that the HQMM with data discretized by MVTF outperforms both HMM and evenly discretized HQMM in temporal efficiency, time (70% faster), computational complexity, and superior predictive performance (4.96% higher). Furthermore, the model captures nonlinear dependencies and hidden correlations unattainable by classical machine learning methods, confirming its quantum information advantage. While this study demonstrates the significant potential of quantum computing in financial forecasting, several limitations

must be acknowledged: (1) data preprocessing has bottlenecks for high-frequency markets, and (2) regulators could leverage quantum-enhanced models for systemic risk monitoring, while financial institutions should establish validation protocols to ensure algorithmic transparency amid quantum adoption. Future research could further enhance model performance by exploring several key directions. These include the advanced design and optimization of quantum circuits, the augmentation of dataset capacity to improve model generalizability, the identification of optimal input cycles tailored to different stock behaviors, the refinement of rule structures within the MVTF model, the development of quantum-specific risk evaluation frameworks, and the standardization of quantum advantage benchmarks in finance. Such advancements would not only contribute to the theoretical evolution of quantum finance but also facilitate its practical implementation within the financial sector. By processing real market data, HQMM enables real-time market regime prediction, automated risk control, and dynamic asset allocation—thus forming an implementable quantum—forming a quantum-enhanced decision system for adaptive portfolio management through accelerated quantum processing.

Author Contributions: Conceptualization, X.S.; methodology, X.S. and J.L.; validation, W.C.; data curation, X.S. and J.L.; writing—original draft preparation, X.S. and J.L.; writing—review and editing, W.C.; visualization, W.C.; supervision, W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data and codes can be obtained from https://github.com/ljy623669155/HQMM-IN-FINANCE.git (accessed on 6 October 2022).

Acknowledgments: The authors would like to thank the editor and anonymous reviewers for their valuable suggestions that helped improve this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HQMM	Hidden quantum Markov model
HMM	Hidden Markov model
AI	Artificial intelligence
ML	Machine learning
GMM	Gaussian mixture model
DA	Distance and accuracy algorithm
MVTF	Means and variances to feature
SZSE	Shenzhen Stock Exchange
SSE	Shanghai Stock Exchange
NC	Natural growth capital
R	Comparative growth rate
FC	Final capital
IC	Initial capital

References

1. Reichenstein, W. Calculating a family's asset mix. *Financ. Serv. Rev.* **1998**, *7*, 195–206. [CrossRef]
2. Malekghaini, N.; Akbari, E.; Salahuddin, M.A.; Limam, N.; Boutaba, R.; Mathieu, B.; Moteau, S.; Tuffin, S. AutoML4ETC: Automated Neural Architecture Search for Real-World Encrypted Traffic Classification. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 2715–2730. [CrossRef]
3. Zhang, B.; Zhuang, Q. Energy-dependent barren plateau in bosonic variational quantum circuits. *Quantum Sci. Technol.* **2025**, *10*, 015009. [CrossRef]

4. Ayyildiz, N.; Iskenderoglu, O. How effective is machine learning in stock market predictions? *Heliyon* **2024**, *10*, e24123. [CrossRef] [PubMed]
5. Mahmoodi, A.; Hashemi, L.; Jasemi, M.; Laliberté, J.; Millar, R.C.; Noshadi, H. A novel approach for candlestick technical analysis using a combination of the support vector machine and particle swarm optimization. *Asian J. Econ. Bank.* **2023**, *7*, 2–24. [CrossRef]
6. Stambaugh, R.F. Predictive regressions. *J. Financ. Econ.* **1999**, *54*, 375–421. [CrossRef]
7. Endri, E.; Kasmir, K.; Syarif, A.D. Delisting sharia stock prediction model based on financial information: Support Vector Machine. *Decis. Sci. Lett.* **2020**, *9*, 207–214. [CrossRef]
8. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
9. Carleo, G.; Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **2017**, *355*, 602–606. [CrossRef]
10. von Lilienfeld, A. Quantum Machine Learning. In *Bulletin of the American Physical Society*; American Physical Society (APS): Washington, DC, USA, 2017; p. B1.001.
11. Yu, L.-H.; Li, X.-Y.; Chen, G.; Zhu, Q.-S.; Li, H.; Yang, G.-W. QUSL: Quantum unsupervised image similarity learning with enhanced performance. *Expert Syst. Appl.* **2024**, *258*, 125112. [CrossRef]
12. Jeswal, S.K.; Chakraverty, S. Recent Developments and Applications in Quantum Neural Network: A Review. *Arch. Comput. Methods Eng.* **2019**, *26*, 793–807. [CrossRef]
13. Bagheri, A.; Mohammadi Peyhani, H.; Akbari, M. Financial forecasting using ANFIS networks with Quantum-behaved Particle Swarm Optimization. *Expert Syst. Appl.* **2014**, *41*, 6235–6250. [CrossRef]
14. Sornette, D. Physics and financial economics (1776–2014): Puzzles, Ising and agent-based models. *Rep. Prog. Phys.* **2014**, *77*, 062001. [CrossRef] [PubMed]
15. Guidolin, M. Markov Switching Models in Empirical Finance. In *Missing Data Methods: Time-Series Methods and Applications*; Drukker, D.M., Ed.; Advances in Econometrics; Emerald Group Publishing Limited: Leeds, UK, 2011; Volume 27, pp. 1–86.
16. Li, X.-Y.; Zhu, Q.-S.; Hu, Y.; Wu, H.; Yang, G.-W.; Yu, L.-H.; Chen, G. A new quantum machine learning algorithm: Split hidden quantum Markov model inspired by quantum conditional master equation. *Quantum* **2024**, *8*, 1232. [CrossRef]
17. Monras, A.; Beige, A.; Wiesner, K. Hidden Quantum Markov Models and non-adaptive read-out of many-body states. *arXiv* **2010**, arXiv:1002.2337. [CrossRef]
18. O'Neill, B.; Barlow, T.M.; Šafránek, D.; Beige, A. Hidden quantum Markov models with one qubit. *AIP Conf. Proc.* **2012**, *1479*, 667–669. [CrossRef]
19. Krogh, A.; Larsson, B.; von Heijne, G.; Sonnhammer, E.L.L. Predicting transmembrane protein topology with a hidden markov model: Application to complete genomes11Edited by F. Cohen. *J. Mol. Biol.* **2001**, *305*, 567–580. [CrossRef]
20. Springer, D.B.; Tarassenko, L.; Clifford, G.D. Logistic Regression-HSMM-Based Heart Sound Segmentation. *IEEE Trans. Biomed. Eng.* **2016**, *63*, 822–832. [CrossRef]
21. Markowitz, H.M. *Portfolio Selection Efficient Diversification of Investments*; Yale University Press: New Haven, CT, USA, 1959.
22. Srinivasan, S.; Gordon, G.; Boots, B. Learning Hidden Quantum Markov Models. *arXiv* **2017**, arXiv:1710.09016.
23. Adhikary, S.; Srinivasan, S.; Boots, B. Learning Quantum Graphical Models using Constrained Gradient Descent on the Stiefel Manifold. *arXiv* **2019**, arXiv:1903.03730.
24. Wen, Z.; Yin, W. A feasible method for optimization with orthogonality constraints. *Math. Program.* **2013**, *142*, 397–434. [CrossRef]
25. Chen, Y.-J.; Chen, Y.-M.; Tsao, S.-T.; Hsieh, S.-F. A novel technical analysis-based method for stock market forecasting. *Soft Comput.* **2018**, *22*, 1295–1312. [CrossRef]
26. Haussmann, U.G.; Sass, J. Optimal Terminal Wealth under Partial Information for HMM Stock Returns. *Contemp. Math.* **2004**, *351*, 171–184.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

mathematics

MDPI

Article

Statistical Data-Generative Machine Learning-Based Credit Card Fraud Detection Systems

Xiaomei Feng and Song-Kyoo Kim *

Faculty of Applied Sciences, Macao Polytechnic University, R. de Luis Gonzaga Gomes, Macao SAR, China
* Correspondence: amang@mpu.edu.mo

Abstract

This study addresses the challenges of data imbalance and missing values in credit card transaction datasets by employing mode-based imputation and various machine learning models. We analyzed two distinct datasets: one consisting of European cardholders and the other from American Express, applying multiple machine learning algorithms, including Artificial Neural Networks, Convolutional Neural Networks, and Gradient Boosted Decision Trees, as well as others. Notably, the Gradient Boosted Decision Tree demonstrated superior predictive performance, with accuracy increasing by 4.53%, reaching 96.92% on the European cardholders dataset. Mode imputation significantly improved data quality, enabling stable and reliable analysis of merged datasets with up to 50% missing values. Hypothesis testing confirmed that the performance of the merged dataset was statistically significant compared to the original datasets. This study highlights the importance of robust data handling techniques in developing effective fraud detection systems, setting the stage for future research on combining different datasets and improving predictive accuracy in the financial sector.

Keywords: credit card fraud; statistical data generation; machine learning; credit prediction; predictive modeling

MSC: 62H30; 62P05; 62P99; 91B08

1. Introduction

The emergence of credit cards in the 1960s has gradually supplanted cash and check transactions, resulting in a considerable and swift rise in both credit card users and transaction volumes, which has consequently triggered a persistent surge in credit card fraud [1]. As a result, financial institutions are increasingly focused on identifying potential borrowers and evaluating the associated credit risks stemming from defaults [2]. Credit card fraud refers to the unauthorized use of a credit card account that happens without the knowledge of the cardholder or the issuing institution regarding a third party engaging in such activities. Fraud perpetrators engage in various unlawful activities which involve acquiring goods or services without making a payment or improperly taking funds from the account of another person. Among these activities are several types of fraud including offline fraud, application fraud, and bankruptcy fraud [3]. The detection and prevention of credit card fraud are vital components of the financial system because they aim to identify and stop all fraudulent transactions that can lead to significant losses for both consumers and institutions. Effective measures in fraud detection are crucial for safeguarding customer trust and ensuring the overall integrity of financial operations [4].

Implementing effective fraud monitoring strategies is a crucial step that can significantly reduce economic losses faced by financial institutions while also greatly enhancing customer trust in their services [5]. These strategies play a vital role in minimizing complaints by proactively addressing fraudulent activities before they escalate. Furthermore, it is essential for financial institutions to rapidly develop and implement robust credit card fraud detection systems that can swiftly identify suspicious transactions, as this will help prevent substantial financial losses associated with these fraudulent actions, which can affect both consumers and businesses. Data integration or generation involves the comprehensive process of consolidating data from various different sources in order to provide users with a coherent and unified view of the information available. Despite the extensive research that has been conducted on the integration of heterogeneous information systems across multiple domains, it remains that most commercial solutions still fall short of achieving full data integration [6]. This persistent issue is compounded by the fact that missing data continues to represent a pervasive challenge that affects many contemporary scientific and engineering domains, resulting in significant obstacles for accurate data analysis and decision-making processes. Missing data continues to represent a pervasive and significant challenge that is prevalent across many contemporary scientific and engineering fields. The occurrence of missing data can lead to the introduction of substantial biases in the information that is processed, thereby resulting in errors during data processing and analysis. This situation ultimately diminishes overall statistical efficiency, impacting the validity and reliability of the results obtained from such analyses. It is crucial for researchers and engineers to address this challenge effectively in order to maintain the integrity of their findings and to ensure that decision-making processes are based on accurate and complete datasets [7].

Generally, missing data is categorized into three main classifications, which include missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR) [8]. Under the MCAR mechanism, data absence is entirely independent of both observed and unobserved variables; thus, missing values do not depend on any specific data value within the dataset. Conversely, the missing at random (MAR) mechanism dictates that the missing data pattern relies exclusively on observed variables, indicating that the missingness can be explained by the available data. This distinction is critical for researchers to select suitable missing data handling techniques, as these choices can significantly affect analysis outcomes and research conclusions. Alternatively, NMAR suggests that the missing data is influenced not only by observed variables but also by the missing values themselves. This dependency introduces significant complexity into the analysis of data completeness, as the reasons for missingness are intrinsically linked to the unobserved data. Simultaneously, the field of credit card fraud detection has witnessed a surge in the adoption of diverse machine learning models, driven by rapid advancements in machine learning techniques. These models are trained and validated using multiple datasets, often sourced from varied origins, to enhance their generalizability and robustness. The primary objective of these techniques is to improve both the accuracy and efficiency of fraud detection systems, thereby providing enhanced support to financial institutions in their efforts to proactively prevent fraudulent transactions and mitigate financial losses. Furthermore, the deployment of these advanced machine learning systems contributes to a more secure and reliable financial ecosystem. Building upon prior studies that have employed extensive credit card datasets for model training, this work proposes integrating two distinct credit card datasets and imputing the resulting missing data to identify potential borrowers and assess their default risk. Specifically, the European Cardholders Dataset and the American Express credit dataset are utilized for data integration, missing data imputation, and subsequent training of machine learning models, as detailed in [3,9]. With the exponential

growth of data generation and the proliferation of data sources, processing and integrating data have become increasingly challenging for businesses.

Companies face challenges that include ensuring data accuracy and timeliness, compounded by the fact that traditional data warehouse approaches can lead to data silos, thereby diminishing the overall effectiveness and benefits of data warehouses [10]. Data holds minimal value for organizations if it cannot be effectively accessed, observed, and utilized. The data integration process facilitates the merging of information from disparate databases and sources, thereby presenting business users with a consolidated and comprehensive view of the data landscape. A robust and well-executed data integration strategy empowers companies to effectively leverage information originating from diverse sources, thereby supporting their overarching business objectives and strategic initiatives. Data integration assumes a pivotal role in contemporary business intelligence systems, impacting not only the precision and timeliness of information but also directly influencing a capacity of a company to make informed decisions in a rapidly evolving market environment. The ability to synthesize and analyze data from various sources enables organizations to respond swiftly to market trends, anticipate customer needs, and maintain a competitive edge. Ultimately, effective data integration is essential for fostering data-driven decision-making and achieving sustainable business success. For efficient business operations, companies must prioritize the design and implementation of robust data integration strategies to better support real-time decision-making processes [11].

Integrating data from diverse business functions, including finance, manufacturing, sales, and marketing, enhances financial transparency and optimizes supply chain operational efficiency. Through comprehensive data integration, businesses can leverage customer behavior information derived from merged datasets to maximize customer satisfaction, foster customer loyalty, and enhance overall profitability [12]. Furthermore, companies can analyze and utilize this integrated data to improve its availability, reliability, and overall quality, thereby ensuring that decision-makers have access to accurate and timely insights. By effectively integrating data assets, businesses are better positioned to leverage their information resources, sustain a competitive advantage in the marketplace, and drive innovation across the organization. This holistic approach to data management and utilization is essential for achieving long-term success and adapting to evolving market dynamics.

This article is further organized into three sections. Section 2 details the preliminaries, focusing on the theoretical underpinnings of this study, including data balancing techniques and various machine learning algorithms previously employed in related research. This section also provides a brief overview of mode-based generative data methods and introduces a novel feature reduction approach utilizing hypothesis testing. The performance of the mode-based generative data method is evaluated using hypothesis testing at different confidence levels. Section 3 presents the experimental results, comparing several machine learning algorithms and feature reduction methods with the mode imputation technique. Finally, Section 4 summarizes the performance comparisons with statistical data generation methods.

2. Preliminaries

The experiments utilizes two publicly available credit card datasets selected from Kaggle. The first selected dataset is the European credit cardholders dataset, which includes 284,807 non-fraud transactions and 492 fraud transactions (i.e., 0.172% out of the total samples) and contains 30 features. The second selected dataset is the American Express credit dataset, which contains 42,579 non-fraud transactions and 2949 fraud transactions (i.e., 6.926% out of the total samples) and includes 18 features [13]. These two publicly

available datasets have been extensively utilized in related research studies [14–16]. During the merging process, all features from the European dataset were retained, while two irrelevant features from the American Express dataset were removed.

2.1. Data Set

The resampling strategy includes both over-sampling and under-sampling techniques, which are commonly employed to address data imbalance issues. Over-sampling through the duplication of minority class samples may lead to overfitting or exacerbate the noise inherent in the dataset [4]. Under-sampling is a technique used to reduce the computational load in order to enhance efficiency. This method can be implemented by either randomly removing samples from the majority class or substituting samples with cluster centroids obtained from a subset of the dataset [4,17]. In this study, we employed random under-sampling techniques on both datasets to mitigate the prevalence of non-fraud samples, with the objective of achieving a nearly equal distribution between fraud and non-fraud classes, each constituting approximately 50%. Utilizing the random under-sampling approach, the European credit cardholders dataset was adjusted to contain 483 non-fraud transactions and 492 fraud transactions (see Figure 1), whereas the American Express credit dataset was modified to include 2966 non-fraud transactions and 2949 fraud transactions (see Figure 2).

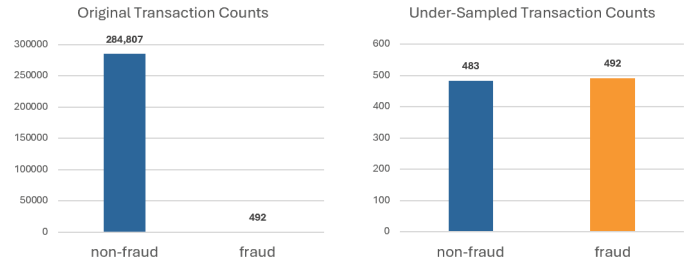


Figure 1. Data balancing for European cardholders dataset [9].

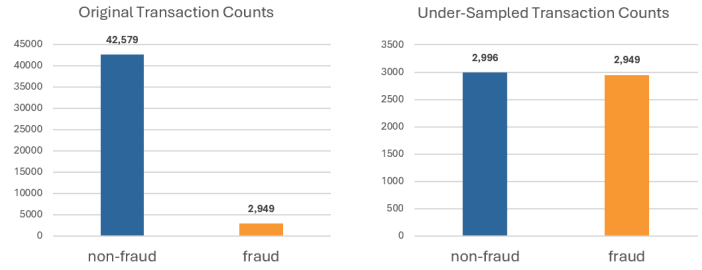


Figure 2. Data balancing for American Express credit dataset [18].

These two selected datasets have been divided into training and testing subsets to train and evaluate various machine learning models. Table 1 presents detailed information on the training and testing datasets for European credit cardholders, including the respective counts for non-fraud and fraud cases. Additionally, it provides the values for both the training and testing datasets of the American Express credit dataset.

Although reducing data samples may negatively impact key performance indicators, such as accuracy, precision, recall, and so on, it is essential to recognize that imbalanced training datasets can result in bias within machine learning algorithms. Therefore, it is imperative to address the issue of imbalance in training datasets to develop robust machine learning systems. In the subsequent phase of our research, we will utilize various machine learning models to train the categorized dataset. These algorithms will be trained on a balanced classification dataset to ensure adequate representation of all classes. This

balanced approach is crucial for enhancing the performance and reliability of the models in predictive tasks.

Table 1. Transaction sample distribution in European and American Express credit datasets.

Dataset	Type	Class 0 (Non-Fraud)	Class 1 (Fraud)
European	Training	393	402
	Testing	90	90
American Express	Training	2341	2391
	Testing	625	558

2.2. Hypothesis Testing

The fundamental principle of a statistical hypothesis test is to determine whether a given data sample is typical or atypical in relation to a specified population, under the assumption that a formulated hypothesis about that population is valid [19]. In scenarios where the sample size is sufficiently large (typically $n \geq 30$), the population standard deviation σ is known, and the underlying data distribution can be assumed to be normal, a Z-test is appropriate. The Z-test facilitates the evaluation of whether the sample mean significantly differs from the hypothesized population mean, thereby providing insights into the validity of the null hypothesis within the context of the study.

In this study, we employed a two-tailed Z-test to assess whether the sample mean significantly differs from the hypothesized population mean. This approach allows us to evaluate deviations in both directions, providing a comprehensive analysis of potential differences. The two-tailed Z-test is utilized to compare the means of two independent samples. The procedure commences with the formulation of hypotheses. The null hypothesis H_0 asserts that there is no significant difference between the two population means, which is mathematically expressed as follows [20]:

$$H_0 : \mu_1 = \mu_2, \tag{1}$$

and, alternatively, the hypothesis H_1 indicates the presence of a significant difference between the two population means, which can be described as follows:

$$H_1 : \mu_1 \neq \mu_2 \quad \text{(for a two-tailed test)}, \tag{2}$$

and the test statistic Z can be calculated as follows:

$$Z = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}, \tag{3}$$

by using an alternative form of the Z-test formula where $\bar{x}_1$ and $\bar{x}_2$ denote the sample means of the first and second groups, respectively; σ_1 and σ_2 are the population standard deviations; while n_1 and n_2 represent the sample sizes. The numerator reflects the difference between the sample means, and the denominator computes the standard error associated with this difference, incorporating the variances of both populations and their sample sizes. Following this computation, the critical value is established based on a predetermined significance level α, typically set at 0.01, 0.05, or 0.10. This critical value is derived from the standard normal distribution table corresponding to the chosen α. For two-tailed tests, it is denoted as $Z_{\alpha/2}$. If the absolute value of the computed Z exceeds the critical value (i.e., $|Z| > Z_{\alpha/2}$), the null hypothesis is rejected. Conversely, if $|Z|$ is less than or equal to the critical value, the null hypothesis is not rejected.

2.3. Mode-Based Generative Data

Mode-based generative data, also known as mode-based data imputation, replaces missing values with the mode, defined as the most frequent value in a variable. This method is a commonly employed technique for addressing missing data [21]. Identifying the mode in this process typically involves utilizing a histogram distribution. A systematic approach to mode imputation, where missing values are replaced with the mode of variables, is presented as a method for handling missing data resulting from merging two datasets. This method proves particularly effective for categorical data by preserving distribution characteristics and maintaining consistency, especially when repeated values are prominent. The process of merging two datasets and performing imputation to generate a complete dataset is illustrated in Figure 3.

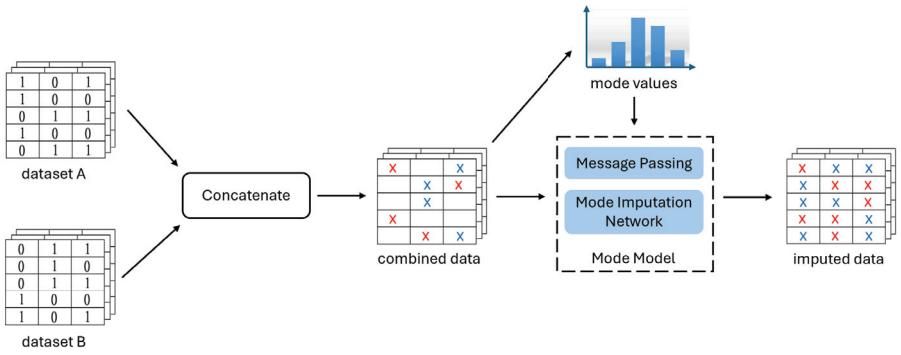


Figure 3. Description of the process for reconstructing missing data sets with mode imputation.

The process of mode imputation is analyzed and documented here through a series of mathematical formulas. Let $\boldsymbol{D} = \{x_1, x_2, \ldots, x_n\}$ be the dataset with missing values. Construct a histogram to visualize the frequency distribution of the observed values in $\boldsymbol{D}$. The histogram will categorize the data into bins. For each bin b_i, calculate the frequency $f(b), b \in \mathbb{B}$, which is of the count of values in bin. The highest frequency of the bin b^* can be found as follows:

$$b^* = \underset{b}{argmax}\left\{^{\exists} b \,|\, f(b), b \in \mathbb{B}\right\}, \tag{4}$$

where $\mathbb{B}$ is the set of the histogram bins and $f(b)$ is the frequency of the bin b. The mean for the value set $\mathbb{V}$ falling within the bin b^* can be determined as follows:

$$\nu = \frac{1}{|\mathbb{V}|} \sum_{x_i \in \mathbb{V}} x_i. \tag{5}$$

The mode-based generative set $\boldsymbol{G}$ which imputes the missing dataset from the dataset $\boldsymbol{D}$ can be determined as follows:

$$\boldsymbol{G} = \left\{^{\forall} g_i \,|\, g_i = x_i \cdot \mathbf{1}_{\{x_i \in V\}} + \nu \cdot \mathbf{1}_{\{x_i \notin V\}}\right\}, \tag{6}$$

and the complete dataset $\tilde{\boldsymbol{D}}$ can be found as follows:

$$\tilde{\boldsymbol{D}} = \boldsymbol{D} \cup \boldsymbol{G}, \tag{7}$$

which includes all original and imputed values. The above process effectively imputes missing values using the mode determined from the histogram of the dataset, thereby enhancing the integrity of the data for further analysis. For each feature within the merged

dataset, imputed values were derived from the same data source, thereby ensuring consistent units post-imputation. Subsequently, non-numeric data underwent label encoding during the preprocessing stage. The European dataset comprises exclusively numeric data, whereas the American Express dataset includes four non-numeric features; label encoding was applied to these, as detailed in Table 2. Prior to model training, `StandardScaler` was systematically deployed for comprehensive data standardization [22]. This essential preprocessing step normalized the features, a measure instrumental in mitigating the influence of varying data scales and thereby ensuring a substantial enhancement in the overall predictive capabilities of the model [23].

Table 2. Encoded features for American Express credit dataset.

Feature	Original	Encoded
gender	F	0
	M	1
owns_car	N	0
	Y	1
owns_house	N	0
	Y	1
occupation_type	Accountants	0
	Cleaning staff	1
	Cooking staff	2
	Core staff	3
	Drivers	4
	HR staff	5
	High skill tech staff	6
	IT staff	7
	Laborers	8
	Low-skill laborers	9
	Managers	10
	Medicine staff	11
	Private service staff	12
	Realty agents	13
	Sales staff	14
	Secretaries	15
	Security staff	16
	Unknown	17
	Waiters/barmen staff	18

2.4. Preliminary Performance Comparison for Various ML Models

For model training and analysis, six notable machine learning models, including both traditional and deep learning approaches, have been chosen. The traditional models incorporate ensemble-based learning techniques. Notably, four of these selected machine learning algorithms have been previously applied to the same dataset, as documented in prior research [24–27]. The following machine learning models are employed in this research:

- **Artificial Neural Networks (ANNs)** [28] are mathematical models that emulate the functioning of the biological brain. They consist of interconnected artificial neurons that process input data through a series of weighted sums and nonlinear decision functions. ANNs are typically organized into layers: an input layer that receives external data, one or more hidden layers that facilitate nonlinear modeling by processing signals between neurons, and an output layer that generates final responses.

- **Convolutional Neural Network (CNN)** [26] represent a deep learning methodology extensively utilized across various domains, including image processing, natural language processing, audio analysis, and time series data. The CNN architecture comprises six distinct layers: the input layer, convolutional layer, pooling layer, fully connected layer, SoftMax/Logic layer, and output layer. Notably, the hidden layers, which share a common structural framework, can accommodate varying numbers of channels within each layer.
- **Gradient Boosted Decision Tree (GBDT)** [25] is an ensemble learning algorithm that constructs a powerful predictive model by iteratively training a series of decision trees. In prior research, GBDT has also been employed as a base learner for fixed-size decision trees, effectively addressing the challenges associated with limited tree depth caused by exponential growth.
- **K-Nearest Neighbor (KNN)** [26] constructs the classifier function by performing voting among its local neighboring data points [27,29,30]. The user specifies the number of neighbors, denoted as k, and the initial selection of neighboring points is made randomly. However, this selection can be refined through iterative evaluation.
- **Long Short-Term Memory (LSTM)** [31] networks represent an advanced class of recurrent neural networks designed to retain sequential data over time. They incorporate gates and a memory cell that capture and store historical trends. Each LSTM consists of multiple cells functioning as modules, where data is conveyed along a transport line from one cell to another. This architecture enables LSTMs to effectively manage long-range dependencies in sequential data.
- **Support Vector Machines (SVMs)** [27] are employed for both classification and regression tasks and are widely recognized for their capacity to delineate optimal decision boundaries between distinct class distributions. However, SVMs tend to demonstrate suboptimal performance when confronted with datasets characterized by imbalanced class distributions, the presence of noise, and the overlap of class samples.

The analysis allows for a detailed comparison of the four evaluation metrics—accuracy, precision, recall, and F1-score—in relation to their respective training times. The following tables present the performance and records of two separate datasets across six algorithms, enabling a better observation and comparison of the effectiveness of merged imputed datasets in further experiments. Table 3 illustrates the performance results of the European cardholders dataset, utilizing the six methodologies previously discussed. The results show that the accuracy ranged from 90% to 94%, with SVM achieving the highest accuracy of 93.91%. All methods executed their processes in under 10 s.

Table 3. Performance results of various machine learning algorithms on the European cardholders dataset [9].

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	Training Time [s]
ANN [28]	90.36	90.40	90.35	0.9035	3.595
CNN [26]	91.37	91.49	91.36	0.9136	9.642
GBDT [25]	92.39	92.61	92.37	0.9237	0.941
KNN [26]	91.88	92.17	91.86	0.9186	0.104
LSTM [31]	92.89	92.97	92.88	0.9289	5.377
SVM [27]	93.91	94.22	93.89	0.9390	0.118

The American Express credit dataset performance results are detailed in Table 4. GBDT achieved the highest accuracy at 97.43%, while other methods ranged from 95% to 98% accuracy. KNN showed the lowest accuracy at 90.60%, alongside the shortest execution

time, approaching 0 s. In contrast, CNN exhibited the longest average execution time, approximately 100 s.

Table 4. Performance results of various machine learning algorithms on the American Express credit dataset [18].

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	Training Time [s]
ANN [28]	96.01	96.00	96.03	0.9601	20.723
CNN [26]	96.62	96.62	96.66	0.9662	97.043
GBDT [25]	97.43	97.48	97.50	0.9750	1.705
KNN [26]	90.60	90.61	90.58	0.9060	0.295
LSTM [31]	95.61	95.60	95.62	0.9600	33.785
SVM [27]	95.40	95.41	95.45	0.9540	2.847

3. Experiment Results

This section employed various statistical methods for imputation and compared their performance outcomes. Mode imputation achieved the highest accuracy among the numerous machine learning algorithms evaluated, with the GBDT algorithm demonstrating the highest accuracy across all results. Consequently, mode imputation was selected as the optimal method for subsequent data imputation in this research. Given that this study combined two entirely distinct features from credit card datasets, the proportion of missing values reached 50% after merging. The imputed values for different features were calculated using the corresponding histograms to determine the mode values, and subsequent imputation was performed on the relevant features. The entire data imputation technique is depicted in Figure 4.

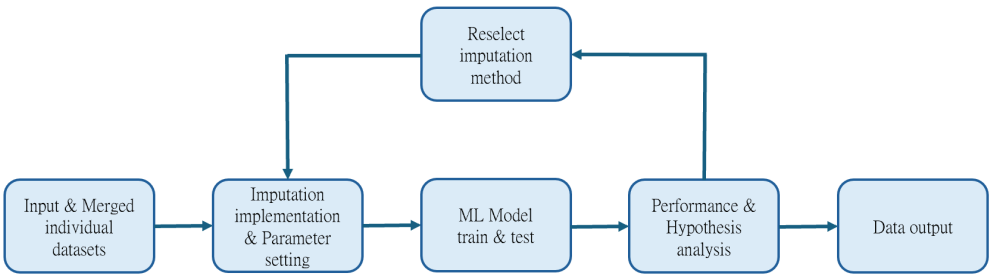


Figure 4. Experimental step diagram.

Initially, the two datasets intended for merging are input. Next, the chosen imputation method is applied to fill in the missing values in the merged dataset, resulting in a complete dataset without any gaps. Following this, model training and prediction are conducted, with the predicted results analyzed through performance evaluation and hypothesis testing. Finally, if the analysis indicates that the performance of the merged dataset is similar to or significantly improved compared to the original dataset, the results and model will be output. Otherwise, the imputation method will be reselected, and relevant parameters will be adjusted to retrain the model.

3.1. Result of the Mode-Based Imputation

A test dataset comprising 46 features and 1316 data samples has been used to perform a performance analysis, and the resulting data was subsequently applied in the hypothesis testing phase. The performance results of the mode-based imputation method are presented in Table 5.

Table 5. Performance results of mode-based generative data.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	Training Time [s]
ANN [28]	95.74	95.74	95.74	0.9574	19.41
CNN [26]	95.45	95.45	95.45	0.9545	216.80
GBDT [25]	96.92	96.95	96.93	0.961	2.23
KNN [26]	90.90	90.91	90.90	0.909	0.41
LSTM [31]	94.64	94.65	94.64	0.946	30.04
SVM [27]	95.52	95.55	95.53	0.955	4.00

It is evident that all key evaluation metrics—including accuracy, precision, and recall—exceed 90%, with F1-score values spanning from 0.90 to 0.97. KNN achieves the lowest metrics at approximately 90.90%, while the other algorithms all exceed 94%. Importantly, GBDT achieved the highest accuracy, precision, and recall, with values of 96.92%, 96.95%, and 96.93%, respectively, and an F1-score of 0.97. Overall, the results from the merged dataset demonstrate stability and reliability, indicating that the imputation process was successful in enhancing data quality. In terms of runtime, although CNN took twice as long to run compared to the time required for the American Express credit dataset, this increase can be attributed to the larger number of parameters involved. In contrast, the runtimes for all other algorithms showed no substantial change. This suggests that despite a 50% increase in data volume, our mode-based imputation model maintained efficient runtime performance, reflecting its effectiveness in handling larger datasets. The performance evaluation of the merged dataset revealed outstanding results from various models in both Receiver Operating Characteristic Area Under Curve (ROC-AUC) and Precision–Recall Curve (PR-AUC) metrics [32,33]. Specifically, GBDT, SVM, CNN, and LSTM consistently achieved ROC-AUC values of 0.99 (see Figure 5), evidencing their high discriminatory capability in differentiating positive from negative samples and capturing complex data patterns. KNN also performed effectively, attaining a ROC-AUC value of 0.97, reflecting strong classification performance.

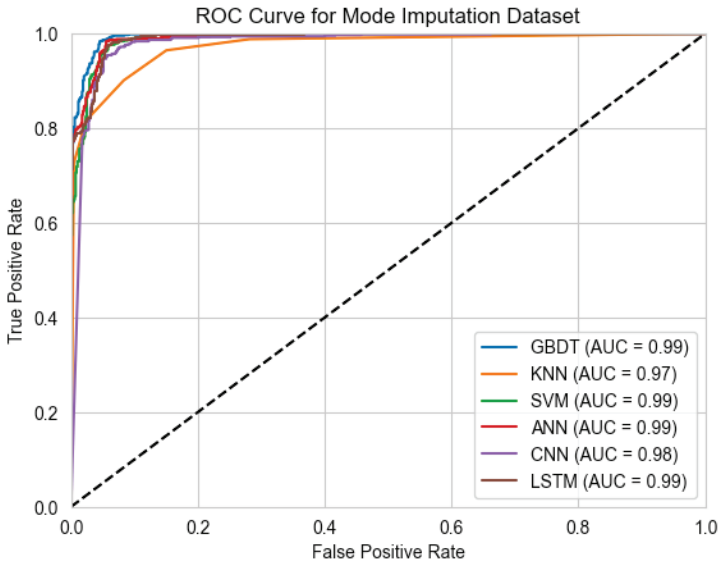


Figure 5. ROC-AUC analysis for mode imputation techniques.

Regarding PR-AUC, the assessed models (e.g., GBDT, SVM, CNN, and LSTM) consistently demonstrated robust performance, with values spanning from 0.96 to 0.99 (as depicted in Figure 6). This range definitively reinforces their notable efficacy in adeptly

managing and predicting outcomes within inherently imbalanced datasets, a critical consideration in real-world applications. While the PR-AUC for KNN registered a slightly reduced value of 0.93, this metric remains within an acceptable and practically viable operational spectrum for classification tasks.

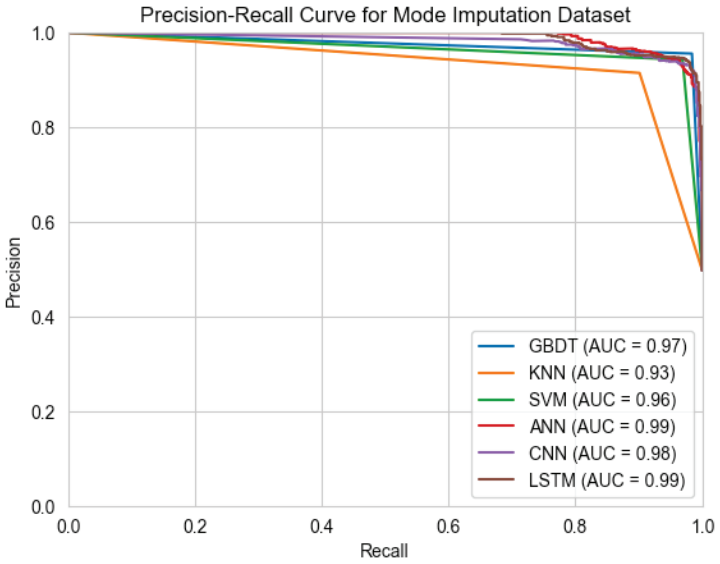


Figure 6. Precision–recall analysis for mode imputation techniques.

The outcomes demonstrate that models from the merged dataset exhibit strong overall accuracy and offer considerable benefits in addressing imbalanced datasets.

3.2. Hypothesis Testing Results

To evaluate the effectiveness of the merged dataset, we employed hypothesis testing. This method systematically evaluates whether the data support a specific hypothesis, allowing for informed conclusions. We used varying significance levels to compare the merged dataset with the original dataset. Compared to the American Express dataset, there is a slight decline in performance. To confirm that the merged dataset is not significantly different from the American Express dataset, we set a significance level of 0.1, which means we are lowering the threshold for rejecting the null hypothesis H_0. We reject H_0 when the p-value is less than 0.1. Table 6 presents the accuracy results of the American Express dataset compared to the merged and imputed dataset. The accuracies of KNN and SVM showed slight improvements, whereas GBDT, ANN, and LSTM experienced minor declines, all of which were less than 1%. However, CNN exhibited a decrease of 1.17%. In the hypothesis testing conducted at $\alpha = 0.1$, which indicates 90% of the confident level, all models showed non-significant changes except for CNN, which demonstrated a significant decline. Therefore, while the null hypothesis H_0 is not accepted for CNN, it is accepted for the remaining algorithms.

To validate the effectiveness of the comparison between the merged dataset and the European dataset, we employed a significance level of 0.01 during hypothesis testing. This was done to confirm that the overall performance of the merged dataset significantly improved compared to the European dataset. This approach ensured that the α value was set at a high standard, allowing us to identify strong evidence that supports the observed effect as statistically significant. Specifically, we reject the null hypothesis H_0 when the p-value is less than 0.01, thereby accepting the alternative hypothesis H_1, which indicates that the performance of the merged dataset is indeed significantly better than that of the European dataset. Table 7 presents the accuracy of the European dataset in comparison to

the merged and imputed dataset. The results were evaluated using the hypothesis testing method, which revealed a significant overall improvement in accuracy for the merged dataset. Specifically, the accuracies of GBDT, SVM, ANN, CNN, and LSTM increased by 4.53%, 1.61%, 5.83%, 4.08%, and 1.75%, respectively. In the hypothesis testing conducted at $\alpha = 0.01$, which indicates 99% of the confident level, all these models demonstrated significant improvements, leading to the acceptance of the alternative hypothesis H_1. However, KNN experienced a slight decline that was not statistically significant, resulting in the rejection of the alternative hypothesis H_1. This indicates that the observed data from the merged dataset shows a statistically significant improvement in accuracy compared to the null hypothesis for the European dataset.

Table 6. Evaluation of mode-based generative data compare to the American Express credit dataset.

Algorithm	Original (%)	Accuracy (%)	p-Value	Significance? ($\alpha = 0.1$)	H_0 Accepted?
GBDT	97.43	96.92	0.276	No	Yes
KNN	90.60	90.90	0.701	No	Yes
SVM	95.40	95.52	0.831	No	Yes
ANN	96.01	95.74	0.620	No	Yes
CNN	96.62	95.45	0.038	Yes	No
LSTM	95.61	94.64	0.111	No	Yes

Table 7. Evaluation of mode-based generative data compare to the the European cardholders dataset.

Algorithm	Original (%)	Accuracy (%)	p-Value	Significance? ($\alpha = 0.01$)	H_1 Accepted?
GBDT	92.39	96.92	<0.001	Yes	Yes
KNN	91.88	90.90	0.208	No	No
SVM	93.91	95.52	0.004	Yes	Yes
ANN	90.36	95.74	<0.001	Yes	Yes
CNN	91.37	95.45	<0.001	Yes	Yes
LSTM	92.89	94.64	0.004	Yes	Yes

The adoption of a stricter significance level (i.e., $\alpha = 0.01$) and a stricter similarity threshold (i.e., $\alpha = 0.1$) for hypothesis testing in this study is supported by several considerations. A primary reason is the critical need to minimize Type I errors, especially pertinent in credit risk assessments where falsely rejecting a valid null hypothesis could lead to unwarranted risks and flawed conclusions. A more stringent significance level thus bolsters the findings' robustness. Furthermore, employing the stricter significance enhances the research outcomes' credibility, implying that the null hypothesis is rejected only with substantial supporting evidence. The use of the stricter significance is also common in diverse financial evaluations, notably in models demanding high precision to curtail erroneous conclusions [34,35].

3.3. Advanced Data-Generative Machine Learning System

In comparison to the original dataset, the merged dataset exhibited consistent performance when evaluated against the American Express dataset, while showing notable improvement relative to the European dataset. This observation not only affirms the efficacy of mode imputation within the experiments but also underscores the success achieved through merging two distinctly different feature datasets. The study employed a merged dataset comprised of 1316 transactions (i.e., x_i), which integrated a total of 46 features (i.e.,

b_i). This comprehensive approach facilitates enhanced data quality, thereby supporting more robust analysis and model training outcomes. The process of mode imputation in the context of merging two datasets for enhancing data integrity has been illustrated on Figure 7.

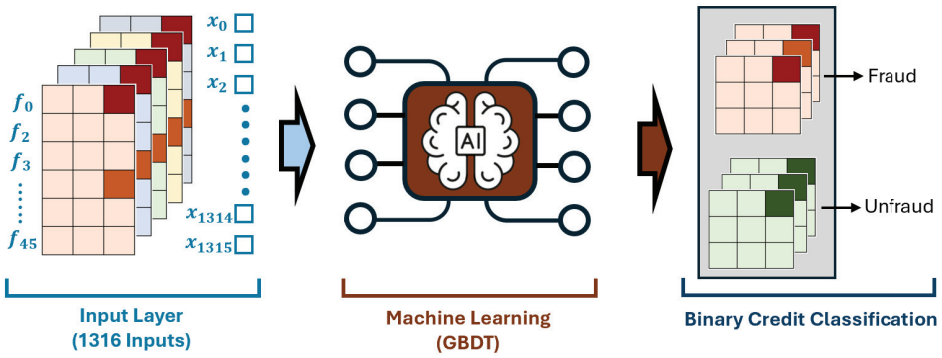


Figure 7. Automated credit card risk detection system.

It showcases how missing values arising from the integration of disparate datasets are systematically filled using the mode, which is the most frequently occurring value in a given data set. This technique is especially beneficial for handling categorical data, as it retains the underlying distribution characteristics and aids in maintaining consistency within the dataset. The figure likely depicts a histogram to visualize the frequency distribution of observed values, identifying the mode effectively. By employing this method, the research aims to create a complete dataset that can be utilized for further analysis, including training models to detect credit card fraud. Ultimately, mode imputation contributes to improving the performance and reliability of machine learning algorithms in fraud detection while ensuring a robust dataset for accurate predictions. All missing values resulting from the merging process were effectively imputed using mode-based data generation. Following the evaluation of model training performance, the GBDT model, which demonstrated the highest accuracy, was chosen for training and prediction. This model successfully identified high-risk transactions among users and generated binary classification outputs, facilitating more accurate risk assessments in financial contexts.

4. Conclusions

This study successfully integrated two distinct credit card datasets and demonstrated the efficacy of mode imputation to address missing data challenges, enhancing both data quality and analytical outcomes. The main contributions include the validation of the GBDT model, which achieved superior predictive performance in identifying high-risk user transactions, as well as the establishment of a robust methodology for dataset integration that can benefit financial institutions. Limitations are present, as this method may primarily apply to specific contexts in the financial sector, making broader generalization difficult. Future investigations can benefit from examining how integrating datasets from various industries influences customer behavior, enabling new insights into fraud detection that transcend conventional frameworks. Investigating alternative imputation methods and their combinations might further enhance data integrity and predictive accuracy. Ultimately, this research paves the way for advancements in fraud detection mechanisms and demonstrates the critical importance of high-quality data in effective financial analysis.

Author Contributions: Writing—draft, X.F.; conceptualization, S.-K.K.; software, X.F.; Visualization, X.F.; writing—revision, S.-K.K.; experiments, X.F.; review, S.-K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Macao Polytechnic University (MPU), under Grant RP/FCA-05/2024.

Data Availability Statement: The datasets for this paper is available on Kaggle repository (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (accessed on 30 June 2025); (Credit Card Fraud Detection), https://www.kaggle.com/datasets/pradip11/amexpert-codelab-2021 (accessed on 30 June 2025); (American Express CodeLab 2021), No DOI available). The merged datasets of imputation are available on GitHub (https://github.com/AliceFeng0417/Merged-credit-datasets (accessed on 30 June 2025)).

Acknowledgments: This paper was revised by using AI/ML-assisted tools. The authors are much in debt to anonymous referees for their careful reviews of our manuscript. Their insightful suggestions led to a notable improvement.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Banker, S.; Dunfield, D.; Huang, A. Neural mechanisms of credit card spending. *Sci. Rep.* **2021**, *11*, 4070. [CrossRef] [PubMed]
2. Tang, Q.; Tong, Z.; Yang, Y. Large portfolio losses in a turbulent market. *Eur. J. Oper. Res.* **2021**, *292*, 755–769. [CrossRef]
3. Makki, S.; Assaghir, Z.; Taher, Y.; Haque, R.; Hacid, M.S.; Zeineddine, H. An Experimental Study with Imbalanced Classification Approaches for Credit Card Fraud Detection. *IEEE Access* **2019**, *7*, 93010–93022. [CrossRef]
4. Ghaleb, F.A.; Saeed, F.; Al-Sarem, M.; Qasem, S.N.; Al-Hadhrami, T. Ensemble Synthesized Minority Oversampling-Based Generative Adversarial Networks and Random Forest Algorithm for Credit Card Fraud Detection. *IEEE Access* **2023**, *11*, 89694–89710. [CrossRef]
5. Tingfei, H.; Guangquan, C.; Kuihua, H. Using Variational Auto Encoding in Credit Card Fraud Detection. *IEEE Access* **2020**, *8*, 149841–149853. [CrossRef]
6. Jens, B.; Felix, N. Data fusion. *ACM Comput. Surv.* **2009**, *41*, 1–41.
7. Huang, S.K.Y.; Song, J. Unsupervised data imputation with multiple importance sampling variational autoencoders. *Sci. Rep.* **2025**, *15*, 3409.
8. Little, R.; Rubin, D. *Statistical Analysis with Missing Data*; Wiley: New York, NY, USA, 2019.
9. Muslim, M.A.; Nikmah, T.L.; Pertiwi, D.A.; Dasril, Y. New Model Combination Meta-learner to Improve Accuracy Prediction P2P Lending with Stacking Ensemble Learning. *Intell. Syst. Appl.* **2023**, *18*, 200–204. [CrossRef]
10. Perez Martinez J. M.; Berlanga R.; Aramburu M. J.; Pedersen T. B. Integrating Data Warehouses with Web Data: A Survey. *IEEE Tran. Know. Data Eng.* **2008**, *20*, 940–955. [CrossRef]
11. Dayal, U.; Castellanos, M.; Simitsis, A.; Wilkinson, K. *Data Integration Flows for Business Intelligence*; EDBT '09; Association for Computing Machinery: New York, NY, USA, 2009; pp. 1–11.
12. Nofal, M.I.; Yusof, Z.M. Integration of Business Intelligence and Enterprise Resource Planning within Organizations. *Procedia Technol.* **2013**, *11*, 658–665. [CrossRef]
13. Feng, X.; Kim, S.K. Novel Machine Learning Based Credit Card Fraud Detection Systems. *Mathematics* **2024**, *12*, 1869. [CrossRef]
14. Rajora, S.; Li, D.L.; Jha, C.; Bharill, N.; Patel, O.P.; Joshi, S.; Puthal, D.; Prasad, M. A Comparative Study of Machine Learning Techniques for Credit Card Fraud Detection Based on Time Variance. In Proceedings of the IEEE Proceedings of SSCI, Bangalore, India, 18–21 November 2018; pp. 1958–1963.
15. Tanouz, D.; Subramanian, R.R.; Eswar, D.; Reddy, G.V.P.; Kumar, A.R.; Praneeth, C.V.N.M. Credit Card Fraud Detection Using Machine Learning. In Proceedings of the IEEE Proceedings of ICICCS, Madurai, India, 6–8 May 2021; pp. 967–972.
16. El hlouli, F.Z.; Riffi, J.; Mahraz, M.A.; El Yahyaouy, A.; Tairi, H. Credit Card Fraud Detection Based on Multilayer Perceptron and Extreme Learning Machine Architectures. In Proceedings of the 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 9–11 June 2020.
17. Fernandez, A.; Garcia, S.; Galar, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer: Cham, Switzerland, 2018.
18. Basak, P. *AmExpert CodeLab 2021: Credit Card Default Risk*; Kaggle: San Francisco, CA, USA, 2021.
19. Emmert-Streib, F.; Dehmer, M. Understanding Statistical Hypothesis Testing: The Logic of Statistical Inference. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 945–961. [CrossRef]
20. Baron, M. *Probability and Statistics for Computer Scientists*; CRC Press: New York, NY, USA, 2019.
21. Batista, G.E.A.P.A.; Monard, M.C. An analysis of four missing data treatment methods for supervised learning. *Appl. Artif. Intell.* **2003**, *17*, 519–533. [CrossRef]

22. Afriyie, J.; Tawiah, K.; Pels, W.; Addai-Henne, S.; Dwamena, H.; Emmanuel, O.; Ayeh, S.; Eshun, J. A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decis. Anal. J.* **2023**, *6*, 100–163. [CrossRef]
23. Scikit-Learn Developers. Sklearn.Preprocessing.StandardScaler. 2024. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (accessed on 30 June 2025).
24. Ileberi, E.; Sun, Y.; Wang, Z. Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost. *IEEE Access* **2021**, *9*, 165286–165294. [CrossRef]
25. Alam, T.M.; Shaukat, K.; Hameed, I.A.; Luo, S.; Sarwar, M.U.; Shabbir, S.; Li, J.; Khushi, M. An Investigation of Credit Card Default Prediction in the Imbalanced Datasets. *IEEE Access* **2020**, *8*, 201173–201198. [CrossRef]
26. Alarfaj, F.K.; Malik, I.; Khan, H.U.; Almusallam, N.; Ramzan, M.; Ahmed, M. Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. *IEEE Access* **2022**, *10*, 39700–39715. [CrossRef]
27. Kalid, S.N.; Ng, K.H.; Tong, G.K.; Khor, K.C. A Multiple Classifiers System for Anomaly Detection in Credit Card Data with Unbalanced and Overlapped Classes. *IEEE Access* **2020**, *8*, 28210–28221. [CrossRef]
28. Nur Ozkan-Gunay, E.; Ozkan, M. Prediction of bank failures in emerging financial markets: An ANN approach. *J. Risk Finan.* **2007**, *8*, 465–480. [CrossRef]
29. Oded Maimon, L.R. *Data Mining and Knowledge Discovery Handbook*; Springer: New York, NY, USA, 2010.
30. Taghizadeh-Mehrjardi, R.; Nabiollahi, K.; Minasny, B.; Triantafilis, J. Comparing data mining classifiers to predict spatial distribution of USDA-family soil groups in Baneh region, Iran. *Geoderma* **2015**, *253–254*, 67–77. [CrossRef]
31. Siami-Namini, S.; Namin, A.S. Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. *arXiv* **2018**, arXiv:1803.06386.
32. AbouGrad, H.; Sankuru, L. Online Banking Fraud Detection Model: Decentralized Machine Learning Framework to Enhance Effectiveness and Compliance with Data Privacy Regulations. *Mathematics* **2025**, *13*, 2110. [CrossRef]
33. Yin, C.; Zhang, S.; Wang, J.; Xiong, N.N. Anomaly Detection Based on Convolutional Recurrent Autoencoder for IoT Time Series. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 112–122. [CrossRef]
34. Toan Duc Le, T.; Ho, V.; Ngoc Anh, N.; Man, L. The Relationship between Working Capital Management and Profitability: Evidence in Viet Nam. *Int. J. Bus. Manag.* **2017**, *12*, 175. [CrossRef]
35. Eling, M.; Schuhmacher, F. Does the choice of performance measure influence the evaluation of hedge funds? *J. Bank. Financ.* **2007**, *31*, 2632–2647. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

mathematics

MDPI

Article

Comparative Analysis of Resampling Techniques for Class Imbalance in Financial Distress Prediction Using XGBoost

Guodong Hou [1,2], Dong Ling Tong [1,*], Soung Yue Liew [1] and Peng Yin Choo [1]

[1] Faculty of Information and Communication Technology, University Tunku Abdul Rahman, Kampar Campus, Kampar 31900, Perak, Malaysia; houguodong@1utar.my (G.H.); syliew@utar.edu.my (S.Y.L.); choopy@utar.edu.my (P.Y.C.)
[2] School of Business, Shandong Agriculture and Engineering University, Jinan 250100, China
* Correspondence: tongdl@utar.edu.my

Abstract

One of the key challenges in financial distress data is class imbalance, where the data are characterized by a highly imbalanced ratio between the number of distressed and non-distressed samples. This study examines eight resampling techniques for improving distress prediction using the XGBoost algorithm. The study was performed on a dataset acquired from the CSMAR database, containing 26,383 firm-quarter samples from 639 Chinese A-share listed companies (2007–2024), with only 12.1% of the cases being distressed. Results show that standard Synthetic Minority Oversampling Technique (SMOTE) enhanced F1-score (up to 0.73) and Matthews Correlation Coefficient (MCC, up to 0.70), while SMOTE-Tomek and Borderline-SMOTE further boosted recall, slightly sacrificing precision. These oversampling and hybrid methods also maintained reasonable computational efficiency. However, Random Undersampling (RUS), though yielding high recall (0.85), suffered from low precision (0.46) and weaker generalization, but was the fastest method. Among all techniques, Bagging-SMOTE achieved balanced performance (AUC 0.96, F1 0.72, PR-AUC 0.80, MCC 0.68) using a minority-to-majority ratio of 0.15, demonstrating that ensemble-based resampling can improve robustness with minimal impact on the original class distribution, albeit with higher computational cost. The compared findings highlight that no single approach fits all use cases, and technique selection should align with specific goals. Techniques favoring recall (e.g., Bagging-SMOTE, SMOTE-Tomek) are suited for early warning, while conservative techniques (e.g., Tomek Links) help reduce false positives in risk-sensitive applications, and efficient methods such as RUS are preferable when computational speed is a priority.

Keywords: class imbalance; SMOTE; ADASYN; Borderline-SMOTE; SMOTE-Tomek; SMOTE-ENN; RUS; Tomek Links; Bagging-SMOTE; XGBoost

MSC: 68T09

1. Introduction

In recent years, the application of machine learning to financial distress prediction has attracted increasing attention, owing to its powerful capabilities in feature extraction and nonlinear modeling, which significantly enhance the ability to capture complex patterns in financial data and improve predictive accuracy. However, a core challenge remains: the severe class imbalance inherent in financial datasets. In practice, distressed samples constitute only a small fraction of all listed companies, causing models to be biased toward

the majority class (non-distressed samples) during training. This often results in high overall accuracy but poor recall for the minority class (distressed samples), which greatly undermines the practical value of predictive systems in financial risk management.

Under such circumstances, classification metrics such as accuracy, recall, and precision often fail to reflect a model's true ability to identify distressed samples. Therefore, effectively addressing the class imbalance problem has become a critical issue in financial distress prediction. Resampling techniques are among the most widely adopted solutions, as they adjust the distribution of training samples to achieve class balance. Commonly used oversampling techniques include Synthetic Minority Oversampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN), and Borderline-SMOTE, as well as hybrid approaches such as SMOTE-Tomek and SMOTE-ENN. Meanwhile, undersampling techniques such as Random Undersampling (RUS) and Tomek Links are also frequently used to reduce the dominance of the majority class. While these resampling techniques have shown advantages in various imbalanced classification tasks, there is still a lack of comprehensive comparative studies that assess their effectiveness and applicability in enhancing model performance for real-world financial distress detection.

Despite the growing body of research on resampling techniques, most existing studies either focus on a limited subset of techniques or rely on simulated or non-financial datasets, which may not fully capture the unique characteristics and challenges of real-world financial data. Furthermore, previous comparative analyses often lack a unified experimental framework, making it difficult to draw consistent and practical conclusions regarding the relative performance of different resampling techniques in financial distress prediction.

To address these gaps, this study presents a systematic and in-depth comparative analysis of a broad range of resampling techniques, including SMOTE, ADASYN, Borderline-SMOTE, SMOTE-Tomek, SMOTE-ENN, Random Undersampling, Tomek Links, and Bagging-SMOTE, using real financial data from publicly listed companies. All methods are evaluated under consistent experimental conditions, with XGBoost employed as the base classifier and a unified evaluation framework applied throughout. This approach enables a fair and comprehensive assessment of the strengths and limitations of each technique in the context of financial distress prediction.

By comparing the performance of these methods across multiple metrics relevant to imbalanced classification such as F1-score, area under the receiver operating characteristic curve (AUC), area under the precision-recall curve (PR-AUC), and Matthews correlation coefficient (MCC), we reveal the effectiveness of different resampling techniques in enhancing the identification of distressed samples and provide practical, evidence-based guidance for practitioners and researchers in the field of financial risk management.

2. Related Works

Class imbalance is a common challenge when working with real-world data, often leading to degraded classification performance. This is because most machine learning algorithms are not inherently designed to handle highly imbalanced datasets effectively. Machine learning models favor the majority class, thereby reducing the accuracy of predictions for the minority class. Song et. al. [1] and Engin [2] highlighted this issue and employed various resampling techniques, including SMOTE and ensemble methods, to reshape the dataset size to achieve class balance, thereby improving model performance.

Resampling techniques can generally be categorized into three major groups: oversampling, undersampling, and hybrid resampling. Each approach seeks to address class imbalance by adjusting the size of the majority or minority class to enhance the model's prediction power in detecting minority class instances.

2.1. Oversampling Techniques

Oversampling techniques address class imbalance by generating synthetic samples for the minority class, thereby improving the performance of machine learning models in detecting rare but critical events.

The commonly used oversampling technique is the Synthetic Minority Oversampling Technique (SMOTE), which generates synthetic samples for the minority class by interpolating new synthetic instances between minority class instances and their nearest neighbors [3]. It has been widely applied in biomedical domains to generate synthetic samples of the minority class in protein-protein interactions [4], breast cancer prediction [5], and medical diagnosis [6]. In some practice studies, Alex and Nayahi [7] reported that SMOTE introduced noise during the process of creating synthetic samples. This could be due to instances that are sparsely distributed, and synthetic samples do not accurately reflect the true characteristics of the minority class.

To alleviate the limitations of SMOTE, Han, Wang, and Mao [8] proposed using Borderline-SMOTE, which focuses on generating samples near the class boundary rather than the entire minority class. It operates on the premise that minority samples far from the class boundary contribute minimally to classification improvement, while those closer to the boundary play a critical role in distinguishing between classes. The effectiveness of Borderline-SMOTE has been reported in fault detection systems [9] and text classification [10], where careful parameter tuning is crucial to prevent misclassification at the class boundaries.

Adaptive Synthetic Sampling (ADASYN) is an advanced oversampling technique that extends SMOTE by adaptively focusing on minority class instances that are harder to learn [11]. The method increases the sampling rate for minority instances that are misclassified or located near the decision boundary, thereby enhancing the classifier's focus on complex regions of the input space. It has been reported to outperform other methods in multiclass imbalanced scenarios when combined with stacking algorithms [12]. Analogous to other oversampling techniques, ADASYN can easily overfit noisy regions if not combined with any other denoising techniques.

2.2. Undersampling Techniques

Unlike oversampling techniques that increase the number of instances for the minority class, undersampling techniques tackle class imbalance by reducing the number of instances in the majority class.

Random Under Sampling (RUS) is an undersampling technique that reduces the size of the majority class by randomly removing instances. Due to its random nature, RUS carries the risk of discarding important information, which is why it is often combined with cleaning strategies. Arifin et al. [13] showed that RUS with decision tree and random forest classifiers performed better than other resampling techniques.

Tomek Links is another widely used undersampling technique, which identifies minority-majority pairs where each is the nearest neighbor of the other, and removing these pairs improves class separation [14]. This technique has been applied in credit card fraud detection [15] and abnormal traffic detection for filtering noise samples in the data [16]. However, it may excessively trim borderline instances to cause an overly aggressive reduction in the dataset's complexity, potentially losing valuable, nuanced patterns that could contribute to a more comprehensive understanding of the predictive model's behavior and performance.

2.3. Hybrid Resampling Techniques

Although oversampling and undersampling techniques are effective for balancing data, relying on a single resampling method has its limitations. To address these shortcomings, hybrid techniques that combine multiple resampling approaches have been proposed.

SMOTE-Tomek, which combines oversampling and undersampling techniques, is widely used in enhancing classifier performance for cancer prediction datasets [17]. It combines SMOTE with Tomek Links to remove noisy majority samples near minority instances, which needed careful validation to avoid overfitting the cleaned subsets.

SMOTE-ENN integrates SMOTE with Edited Nearest Neighbors (ENN) to delete misclassified majority samples post-oversampling, refining decision boundaries. It has been used in medical datasets such as missed abortion diagnosis to enhance the model's classification accuracy [18]. Similar to most of the undersampling techniques, ENN may remove useful samples in its aggressive pruning process.

2.4. Prediction Models with Built-In Resampling Techniques

A major challenge in hybrid resampling techniques is maintaining a balance between enhancing minority class representation and preserving the overall data quality. This enhancement is typically achieved by reducing the number of majority class samples and/or increasing instances of the minority class. However, excessive pruning or over-generating samples can distort the original data distribution, potentially leading to inaccurate predictions.

Recent studies have demonstrated the effectiveness of XGBoost in handling the class imbalance issue of financial distress scenarios [19,20]. XGBoost incorporates cost-sensitive learning through adjustable weights for minority class instances, optimizing prediction. It employs regularization techniques to prevent overfitting, making it suitable for high-dimensional financial datasets [21].

RUSBoost, which combines Random Undersampling (RUS) with boosting algorithms, was also explored in various application scenarios [22]. It reduces the size of the majority class to balance the dataset, mitigating bias toward majority classes and boosting iteratively trained weak classifiers, emphasizing misclassified instances in subsequent iterations.

2.5. Summary

In summary, oversampling techniques such as SMOTE and ADASYN have proven effective in improving minority class detection, particularly in dense regions of the minority class [23]. However, in financial distress prediction, where distressed companies may be highly heterogeneous and sparsely distributed, these methods risk introducing synthetic noise or generating unrealistic samples [24], potentially distorting the true risk profile of companies. Undersampling techniques such as Tomek Links can enhance class separation and reduce overlap, but they may also discard valuable information from the majority class, which is especially problematic in financial datasets where each observation may carry unique, informative signals about corporate health.

Hybrid techniques, such as SMOTE-Tomek and SMOTE-ENN, often outperform individual methods by combining data synthesis with noise reduction [25]. Nevertheless, these approaches can lead to over-pruning, inadvertently removing borderline or informative samples along with noise, which may reduce the model's ability to capture subtle patterns critical for early warning in financial distress scenarios. Moreover, the effectiveness and side effects of these resampling techniques are highly dependent on dataset characteristics, such as the degree of imbalance, feature distribution, and the presence of outliers, as well as on computational resources, an important consideration for large-scale financial applications.

Previous comparative studies have often focused on general classification tasks or biomedical datasets, with limited attention to the unique challenges posed by financial distress prediction. These limitations underscore the need for a systematic evaluation of resampling techniques in the context of real-world financial datasets, considering both predictive performance and practical trade-offs.

This study aims to address the identified research gaps by critically evaluating the effectiveness and limitations of major resampling techniques on a large, real-world financial dataset. We utilize a comprehensive set of financial indicators, encompassing both general financial ratios and bankruptcy-related features, to ensure robust analysis. Additionally, our study considers computational efficiency and discusses the practical implications of different resampling techniques for financial risk management.

3. Materials and Methods

3.1. Data

The data used in this study were obtained from the China Stock Market & Accounting Research (CSMAR) database. It comprised quarterly financial data from 639 Chinese A-share listed companies from 2007 to 2024, resulting in a total of 26,383 firm-quarter samples with 368 variables. The data are highly imbalanced, with 3181 samples (12.1%) labeled as Special Treated (ST), and 23,202 samples (87.9%) labeled as non-ST (see Figure 1).

Table 1 summarizes the dataset, which includes two subsets: Financial Indicators and Bankruptcy Reorganization. Detailed descriptions are available in CSMAR. The financial indicators span various dimensions, while the bankruptcy reorganization subset adds variables related to business risk.

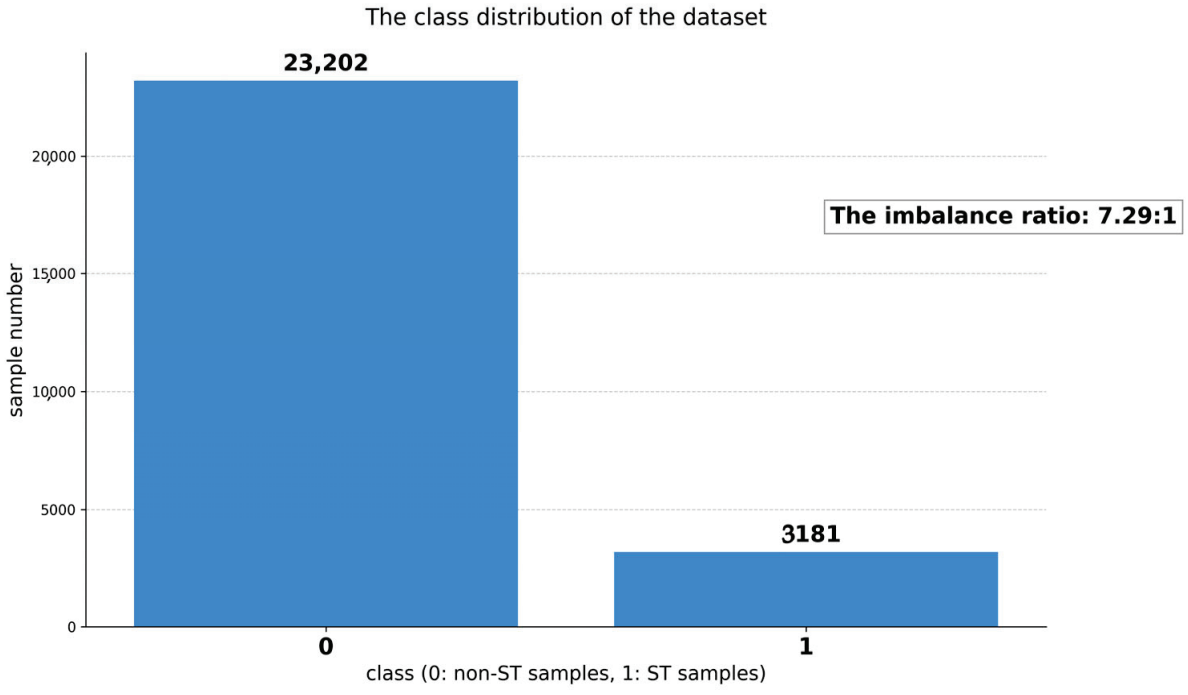


Figure 1. Sample distribution.

Table 1. The dataset components.

Set	Category	Brief Description
Financial Indicators	Solvency	Ability to repay short- and long-term debt using enterprise assets.
	Disclosed Index	Indicators directly reported to reflect company operations.
	Ratio Structure	Financial structure based on the proportions of key financial indicators.
	Operating Capacity	Efficiency in utilizing assets for business operations.
	Earning Capacity	Ability to generate profits.
	Cash Flow	Cash-related indicators derived from financial statement ratios.
	Risk Level	Risk of financial instability due to weak structure or poor financing practices.
	Growth Capability	Potential for future expansion and performance improvement.
	Index per Share	Financial condition measured on a per-share basis.
	Relative Value Index	Derived from comparisons among related indicators.
	Dividend Distribution	Metrics derived from comparisons among related financial indicators.
Bankruptcy Reorganization	Business Risk	Risk profile of bankrupt or restructured listed firms.

3.2. Experimental Design

Figure 2 illustrates the experimental workflow of this study. We first normalize the original dataset using min-max normalization. We then partition the data into training and test sets with a stratified split ratio of 70% for training and 30% for testing. To systematically investigate the impact of class imbalance handling, we compare eight resampling techniques, each applying exclusively to the training data to prevent data leakage. We then use each balanced training set to train the XGBoost model, employing 5-fold cross-validation to mitigate overfitting. Finally, we evaluate each trained XGBoost model on the untouched test set to assess generalization ability and robustness to unseen data.

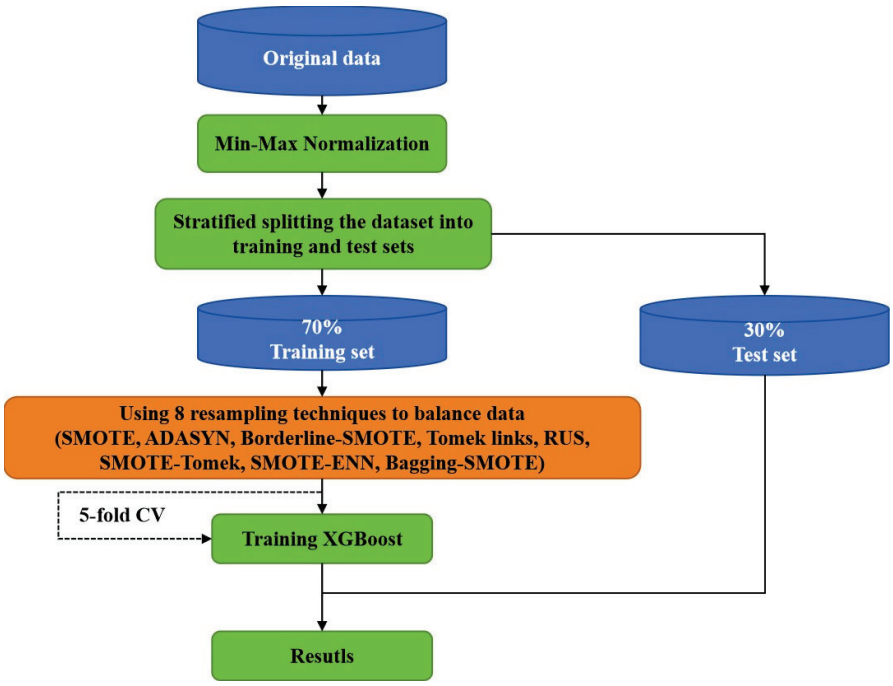


Figure 2. Experimental workflow for the study.

3.2.1. Data Normalization

Min-max normalization was selected in this study to rescale feature values because it preserves the underlying distribution and patterns of the data. Additionally, since the base classifier used (XGBoost) does not assume normality, min-max scaling is a suitable choice. Equation (1) presents the mathematical expression of this method.

$$\text{Min-Max normalization, } X' = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Given a feature X consisting of value $\{X_1,\ X_2,\ \ldots,\ X_i\}$, where i is the total number of observations in the feature, the normalization process involves subtracting the minimum value of X from each value and then dividing by the range. The resulting normalized values retain the original distribution shape but are confined within the range of [0, 1].

3.2.2. Stratified Splitting of the Dataset

To ensure a fair and representative evaluation, we employ stratified splitting to divide the dataset into training and test sets. Stratified splitting preserves the original class distribution in both subsets, which is especially important for imbalanced classification tasks. Importantly, all subsequent data preprocessing steps, including resampling techniques such as oversampling, undersampling, and hybrid resampling, were performed exclusively on the training set. The test set remains completely isolated and was not used in any way during resampling or model training. This protocol effectively prevents data leakage, ensuring that the evaluation metrics reflect the model's true generalization ability on unseen data.

3.2.3. Resampling and Hybrid Techniques

(1) Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is an oversampling technique that creates new instances through linear interpolation between a given minority sample and its k-nearest minority class neighbors. The default number of k neighbors in the standard SMOTE algorithm is 5.

The process involves selecting a minority class sample x_i, identifying one of its nearest neighbors x_j, and generating a synthetic sample x_{synth} as follows:

$$\mathrm{x_{synth}} = \mathrm{x_i} + \lambda \cdot \left(\mathrm{x_j} - \mathrm{x_i}\right) \tag{2}$$

where x_i, and x_j are minority class neighbors, and λ is a random weight $\in$ [0, 1]. This operation effectively generates new samples along the line segments connecting minority instances, which helps to generalize the minority region and reduce model bias toward the majority class.

(2) Borderline-SMOTE

Borderline-SMOTE is a supervised oversampling method that improves upon the original SMOTE algorithm by focusing specifically on minority class instances that lie near the decision boundary, where misclassification is more likely. The technique starts by analyzing the k-nearest neighbors of each minority class instance. Based on the distribution of class labels within this neighborhood, each instance is categorized as safe, noisy, or dangerous. Only those identified as dangerous, which are surrounded by a mix of majority and minority class neighbors, are selected for oversampling.

For a given dangerous minority instance x_i and its k-nearest minority neighbors $\{k_1, k_2, \ldots,\ k_j\}$, the SMOTE interpolation process generates synthetic samples by combining x_i with a selected neighbor k_j. The synthetic instance is created using a linear

interpolation between the two points based on their distance. The mathematical representation is given as follows:

$$x_{synth} = x_i + \lambda \cdot (k_j - x_i) \tag{3}$$

where λ is a random weight $\in [0, 1]$.

(3) Adaptive Synthetic Sampling (ADASYN)

ADASYN is an extension technique of SMOTE that focuses on minority class instances that are harder to learn. It begins by computing the ratio of majority class neighbors within the k-nearest neighborhood of each minority sample. A higher ratio indicates greater difficulty in learning that instance. ADASYN assigns a higher weight to such samples when generating synthetic data. Formally, for each minority sample x_i, the number of synthetic samples to be generated is proportional to its local density measure. Equation (4) is the mathematical expression of this method.

$$g_i = \frac{r_i}{\sum_{j=1}^{n} r_j} \tag{4}$$

where r_i is the ratio of the majority class examples to x_i, and n is the total number of observations in the minority class.

New instances are generated using the same interpolation mechanism as in SMOTE. By concentrating on more challenging areas of the feature space, ADASYN dynamically shifts the decision boundary and enhances the learning process.

(4) Random Undersampling (RUS)

RUS is an undersampling technique that reduces the number of majority class instances by randomly removing samples, thereby balancing the class distribution in the dataset.

The core assumption of RUS is that the majority class contains redundant or less informative instances that can be discarded without significantly compromising model performance. However, due to the random nature of instance removal, there is a potential risk of discarding important majority examples and thus losing valuable information.

(5) Tomek Links

Tomek Links is an undersampling technique that finds pairs of nearest-neighbor instances from different classes. When such pairs are found, instances of the majority class will be removed, thereby refining the decision boundary and reducing overlap between the two classes. Equation (5) shows the mathematical representation of this method.

$$NN(x_i) = x_j \wedge NN(x_j) = x_i \wedge y_i \neq y_j \tag{5}$$

Two instances (x_i,x_j) form a Tomek Link if they are nearest neighbors ($NN(x)$) from opposite classes (y_i,y_j).

(6) SMOTE-Tomek

SMOTE-Tomek is a two-stage hybrid technique that integrates SMOTE oversampling with Tomek Links undersampling [26]. Firstly, SMOTE generates synthetic minority class samples using interpolation between existing instances and their k-nearest neighbors (see Equation (2)). In the latter stage, Tomek Links are identified within the augmented dataset to remove nearest-neighbor instances from the majority class (see Equation (5)).

The combination of oversampling through SMOTE and data cleaning via Tomek Links enables SMOTE-Tomek to enhance both class balance and the discriminative quality of the feature space.

(7) SMOTE-Edited Nearest Neighbors (SMOTE-ENN)

SMOTE-ENN is a hybrid resampling technique that integrates SMOTE with the ENN method [14]. This approach aims to address class imbalance and to enhance the quality of the training dataset by removing noisy or ambiguous instances.

In the first step, SMOTE generates synthetic instances for the minority class in the dataset. The ENN algorithm is then applied to remove any instance whose class label differs from the majority class of its k-nearest neighbors. This rule applies to both original and synthetic examples, enabling the algorithm to eliminate overlapping or noisy points from both classes. By combining data augmentation and noise filtering, SMOTE-ENN improves class separability and enhances generalization performance.

(8) Bagging-SMOTE

Bagging-SMOTE is a prediction model that combines the strengths of bootstrap aggregating and SMOTE to improve classification performance on imbalanced datasets. The training data were first randomly split into K bootstrap folds (with replacement). The SMOTE was then applied to each bootstrap fold to balance the class distribution. In the Bagging-SMOTE framework, we selected XGBoost as the base classifier.

Table 2 summarizes the resampling techniques used in this study.

Table 2. Summary of resampling and hybrid techniques.

Technique	Type	Description
SMOTE	Oversampling	Generates synthetic minority class samples via linear interpolation between neighbors.
ADASYN	Oversampling	Focuses synthesis on difficult minority samples by weighting based on local imbalance.
Borderline-SMOTE	Oversampling	Specifically, it oversamples minority instances near the decision boundary.
Random Under	Undersampling	Randomly reduces the majority class samples to balance the distribution.
Tomek Links	Undersampling	Removes overlapping majority class samples to improve class separation.
SMOTE-Tomek	Hybrid	Apply SMOTE, then clean the noise with Tomek Links.
SMOTE-ENN	Hybrid	Generates synthetic samples and then removes misclassified instances.
Bagging-SMOTE	Hybrid	Stratify data with sampling by replacement, then apply SMOTE in each stratified set of data.

3.2.4. XGBoost Prediction Model

XGBoost is a powerful gradient boosted trees algorithm known for its effectiveness in handling complex datasets. We select it as the base classifier in this study due to its strong performance in tabular data modeling, robustness to multicollinearity, and ability to handle missing values—characteristics well-suited to financial datasets with high dimensionality and sparsity. In addition, XGBoost offers effective regularization to prevent overfitting, supports various objective functions, and provides efficient, scalable implementations that are widely adopted in both academic research and industry. These advantages make it a

practical and reliable choice for our analysis focused on evaluating resampling techniques under class imbalance.

During model training, it begins with an initial prediction based on the distribution of the target variable. We then train the subsequent trees to correct the errors of the previous model by focusing on the misclassified samples. Each new tree works to refine the predictions of the previous model by re-predicting the incorrectly classified samples. This iterative process continues until either the predefined number of trees is built or the number of unchanged predictions reaches a specified threshold.

To mitigate overfitting, the experiment employs 5-fold cross-validation (CV) during training. In each iteration, it trains the model on four folds of the data, while the remaining fold is used to validate its predictive performance. This process is repeated until each fold has served once as the validation set. The experiment uses average performance across all folds to assess the model's generalization ability. Table 3 shows the used hyperparameters of the model.

Table 3. Hyperparameter in XGBoost.

Parameter	Value	Descriptions
random_state	42	To ensure the reproducibility of experimental results
n_estimators	300	The maximum number of trees
eval_metric	Logloss, AUC	To evaluate the performance of tree prediction
early_stopping_rounds	20	Early stopping to prevent model overfit

To comprehensively assess the predictive performance of the XGBoost classifier, we employ a range of evaluation metrics, including accuracy, precision, recall, F1-score, area under the receiver operating characteristic curve (AUC-ROC), area under the precision-recall curve (PR-AUC), and Matthews correlation coefficient (MCC). Equations (6)–(11) present the mathematical representation of these metrics.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

where,

TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

$$\text{MCC} = \frac{(\text{TP} \times \text{TN}) - (\text{FP} \times \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{10}$$

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}) \text{d}(\text{FPR}) \tag{11}$$

where,

TPR (True Positive Rate) is also known as sensitivity or recall.
FPR (False Positive Rate) is defined as $\frac{FP}{FP+TN}$.

4. Results

To preserve the original class imbalance ratio, we stratified the dataset into training and test subsets. Figure 3 illustrates the class distribution of the entire dataset, training, and test subsets. The proportion of the minority class (i.e., Class 1, representing distressed samples) is consistently 12.1%, while the majority class (i.e., Class 0, non-distressed samples) accounts for 87.9% across all splits.

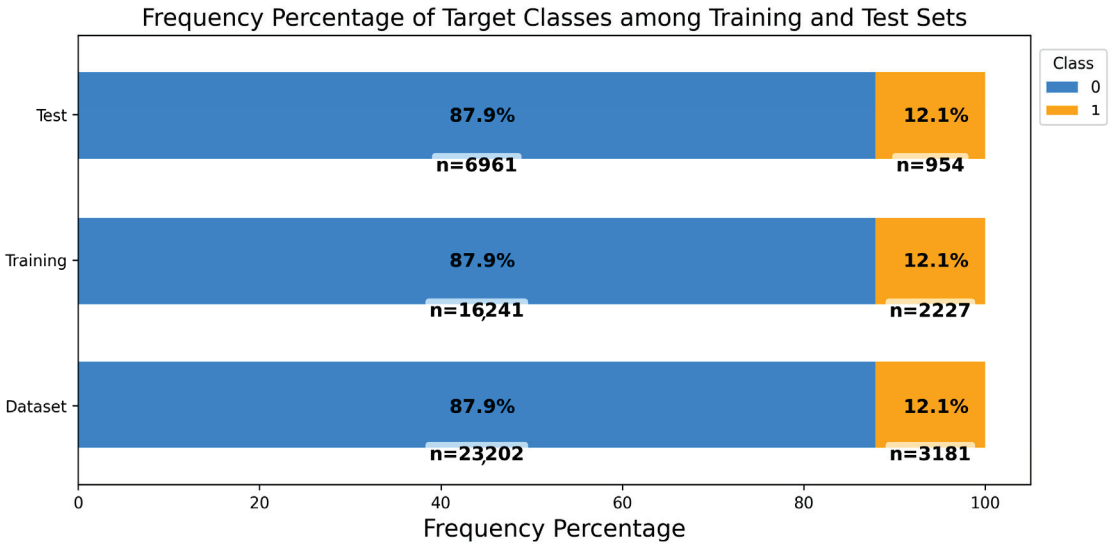


Figure 3. Class distribution among training and test sets.

4.1. Impact of Resampling Techniques Toward the Training Set

To address class imbalance, we applied eight resampling techniques to the training subset. Figure 4 compares the class distribution of the original training set against the effects of the eight resampling methods on the minority and majority classes.

SMOTE (see Figure 4b), ADASYN (see Figure 4c), and Borderline-SMOTE (see Figure 4d) generate synthetic minority samples to balance the dataset, with ADASYN focusing more on difficult-to-learn regions. SMOTE-Tomek (see Figure 4g) and SMOTE-ENN (see Figure 4h) combine oversampling with cleaning methods by removing noisy or overlapping majority class samples, resulting in clearer class boundaries. RUS (see Figure 4e) reduces the number of majority samples, leading to a more compact but potentially underrepresentative distribution. Tomek Links (see Figure 4f) remove borderline majority class samples that overlap or are close to the minority class, resulting in a cleaner class boundary. As shown by the purple '×' markers, a small subset of the majority class is eliminated, leading to minimal distributional shift while retaining the overall structure of the data. Bagging-SMOTE (see Figure 4i) produces a similarly distributed training set in each fold, following a sampling strategy of 0.15, where synthetic minority class samples are generated to reach approximately 15% of the majority class size. The synthetic minority samples (green triangles) enhance minority representation in overlapping areas of the feature space. These visualizations collectively demonstrate how each technique reshapes the data space to address class imbalance and enhance classifier learning.

Table 4 summarizes the class distribution comparison before and after applying the resampling techniques.

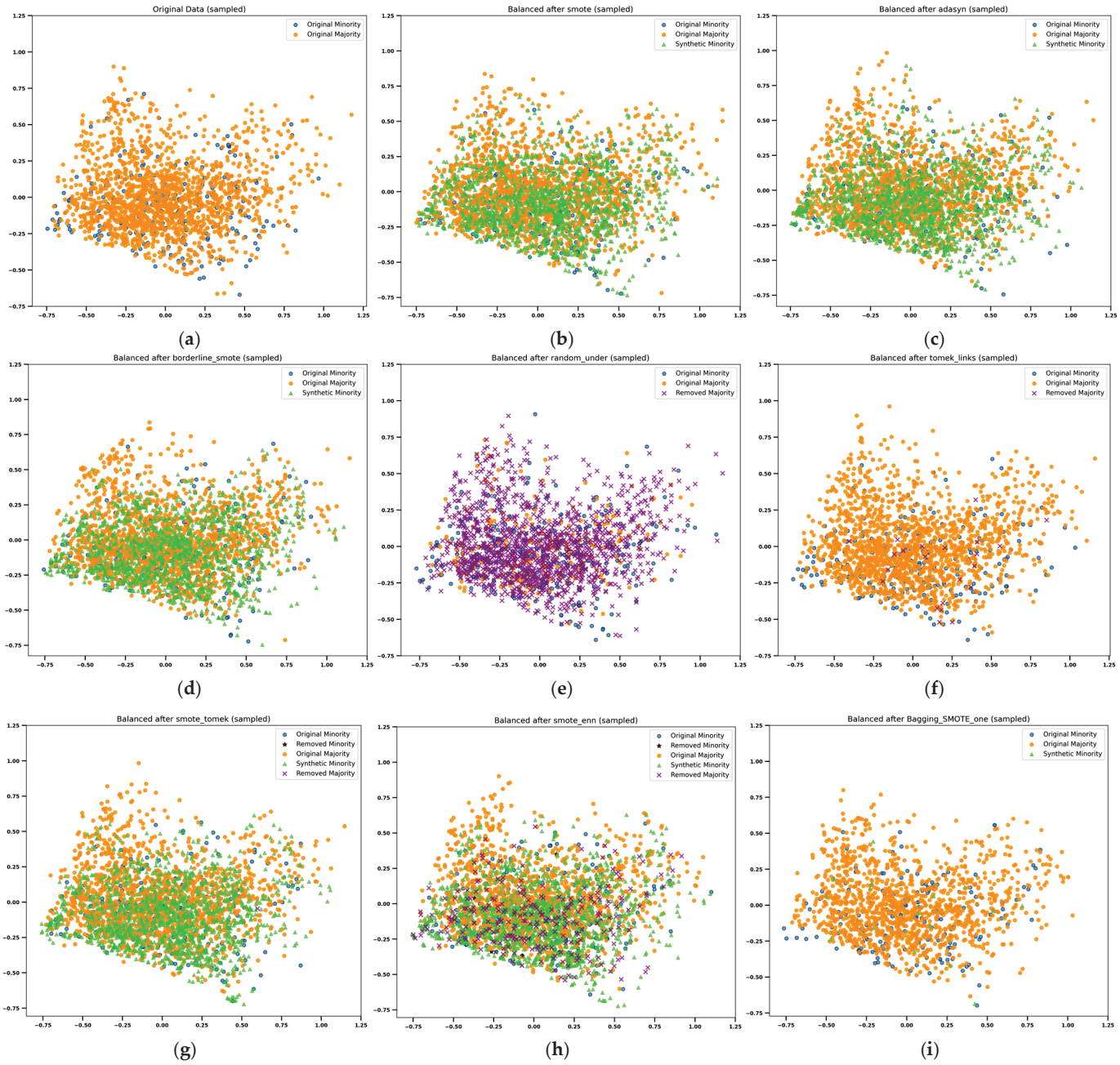


Figure 4. Sample distribution after resampling. (**a**) Original training data; (**b**) SMOTE; (**c**) ADASYN; (**d**) Borderline-SMOTE; (**e**) RUS; (**f**) Tomek Links; (**g**) SMOTE-Tomek; (**h**) SMOTE-ENN; (**i**) Bagging-SMOTE. Legends and marker styles are unified: blue circles = original minority, orange dots = original majority, green triangles = synthetic minority, purple crosses = removed majority, red stars = removed minority. Only 10% of samples are shown for clarity.

Oversampling techniques such as SMOTE, ADASYN, and Borderline-SMOTE elevated the number of distressed (minority) samples from 2227 to 16,241, thereby achieving a fully balanced class distribution with a 1:1 ratio. In contrast, undersampling techniques such as RUS addressed the imbalance by reducing the number of non-distressed (majority) samples down to 2227, equalizing the class frequencies at the cost of discarding a substantial portion of the majority class data.

Hybrid approaches implemented more nuanced balancing strategies. For instance, Tomek Links applied a light filtering mechanism, resulting in a slight reduction in majority class samples (from 16,241 to 16,021), while keeping the minority class unchanged, maintaining a high imbalance ratio of approximately 7.19:1. This reflects its design, which

aims to eliminate overlapping or borderline instances rather than adjust class frequencies aggressively.

Table 4. Class distribution comparison before and after resampling.

Methods	Class Distribution		
	Class 0	**Class 1**	**Class Ratio (0:1)**
Original data	16,241	2227	7.29:1
SMOTE	16,241	16,241	1:1
ADASYN	16,241	16,273	1:1
Borderline-SMOTE	16,241	16,241	1:1
RUS	2227	2227	1:1
Tomek Links	16,021	2227	7.19:1
SMOTE-Tomek	16,229	16,229	1:1
SMOTE-ENN	13,702	16,018	0.86:1
Bagging-SMOTE	16,241	2436	6.67:1

Class 0 denotes non-distressed samples, and Class 1 denotes distressed samples.

Other hybrid techniques, such as SMOTE-Tomek, combined synthetic sample generation with instance filtering to achieve perfect balance, increasing the minority class to 16,229 while slightly reducing the majority class to a comparable number. SMOTE-ENN, which applies to a more aggressive noise-filtering mechanism after oversampling, led to an imbalanced but cleaner dataset, yielding 16,018 distressed and 13,702 non-distressed samples, with a class ratio of 0.86:1. This filtering step aims to remove noisy or ambiguous instances, improving data quality and potentially enhancing model generalization.

Bagging-SMOTE integrates bootstrapped training with SMOTE within an ensemble framework, maintaining consistent resampling outcomes across folds. In each fold of the cross-validation process, four estimators were used for training and one for validation. The majority class in the training estimators comprised 16,241 samples, and SMOTE synthetically increased the minority class samples to about 2436 based on the defined sampling strategy of 0.15. Bagging-SMOTE effectively reshapes the data space by generating synthetic minority instances within the existing data manifold, mitigating class imbalance while maintaining diversity across ensemble folds.

Overall, the visualization and tabulated results highlight the diverse impacts of various resampling techniques on class distribution. These adjustments are critical for addressing the severe class imbalance inherent in financial distress prediction tasks. By balancing the dataset, either through oversampling, undersampling, or hybrid resampling, these techniques enhance the model's capacity to detect distressed samples, which would otherwise be underrepresented and potentially overlooked during the learning process due to the overwhelming dominance of the majority class.

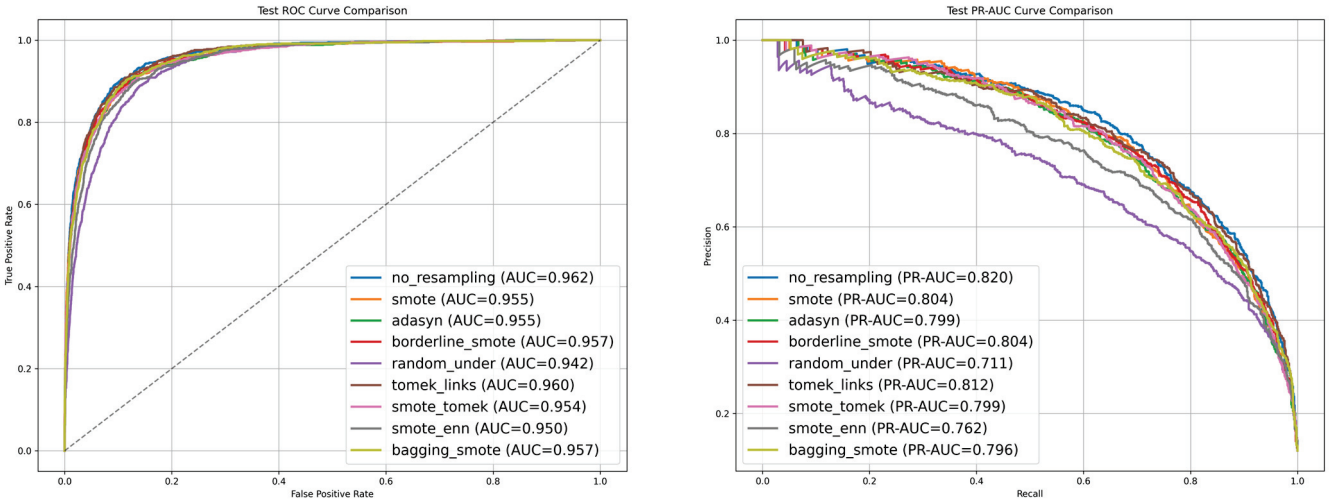
4.2. Impact of Resampling Techniques Toward Prediction Model

To evaluate the effectiveness of resampling techniques in enhancing the model's prediction performance of predicting financial distress, we used the original imbalanced test data to test. Table 5 and Figure 5 summarize the prediction performance of XGBoost models combined with various resampling techniques, including no-resampling, SMOTE, Borderline-SMOTE, ADASYN, SMOTE-ENN, SMOTE-Tomek, Tomek Links, RUS, and Bagging-SMOTE. The examined metrics are accuracy, precision, recall, F1-score, the area under the ROC curve (AUC), the area under the precision-recall curve (PR-AUC), the Matthews Correlation Coefficient (MCC), sampling time, and training time.

Table 5. Classification results on the test set and time costs.

Methods	Accuracy	Precision	Recall	F1	AUC	PR-AUC	MCC	Sampling Time (s)	Training Time (s)
Original data	0.94	**0.85**	0.60	0.71	0.96	**0.82**	0.68	-	42.6
SMOTE	0.94	0.77	0.69	**0.73**	0.96	0.80	**0.70**	2.4	70.0
ADASYN	0.94	0.76	0.68	0.72	0.96	0.80	0.68	2.3	70.4
Borderline-SMOTE	0.94	0.77	0.69	**0.73**	0.96	0.80	0.69	2.4	71.4
RUS	0.87	0.48	**0.88**	0.62	0.94	0.71	0.58	**1.6**	**14.9**
Tomek Links	0.94	0.82	0.61	0.70	0.96	0.81	0.68	2.7	41.0
SMOTE-Tomek	0.94	0.75	0.69	0.72	0.95	0.80	0.69	5.9	68.5
SMOTE-ENN	0.92	0.63	0.77	0.69	0.95	0.76	0.65	6.1	58.1
Bagging-SMOTE	0.93	0.74	0.69	0.72	0.96	0.80	0.68	-	400.3

Values in boldface indicate the best result from the metrics.



(**a**) (**b**)

Figure 5. Comparison of ROC and PR-AUC curves. (**a**) ROC curve; (**b**) PR-AUC curve.

In the task of financial distress prediction, resampling techniques significantly influence model performance by mediating the trade-off between precision and recall. The benchmark model without any resampling achieves the highest precision (0.85), indicating that most samples it predicts as distressed are indeed distressed. However, its relatively low recall (0.60) indicates that it fails to identify a considerable number of actual distressed cases, revealing a sensitivity issue likely caused by class imbalance. Despite its simplicity and fast processing (training time: 42.6 s), the original model may not be reliable for early warning systems due to under-identification of distressed observations.

In contrast, RUS achieves the highest recall (0.88), demonstrating its effectiveness in detecting distressed samples, but this comes at the cost of significantly reduced precision (0.48) and overall accuracy (0.87). This pattern suggests that while undersampling improves minority class detection, it does so by introducing more false positives. This occurs because RUS balances the classes by aggressively reducing the majority class, resulting in the model learning comprehensive patterns of the minority class but only limited information from the majority class. However, RUS is the most computationally efficient among all methods (training time: 14.9 s), making it suitable for scenarios where detection sensitivity is prioritized and resources are limited. Tomek Links offers a moderate recall of 0.61 and a precision of 0.82, resulting in an F1-score of 0.70 and MCC of 0.68. Its runtime remains low,

requiring 2.7 s for sampling and only 41 s for training, making it a time-efficient option with strong precision.

Among oversampling techniques, SMOTE, ADASYN, and Borderline-SMOTE present a more balanced and stable performance. These techniques improve recall to approximately 0.68–0.69 while maintaining moderate precision levels (0.76–0.77), resulting in relatively strong F1-scores (0.72–0.73). Additionally, their AUC values (0.96) and PR-AUC values (0.80) remain consistently high, indicating strong discriminative power across both classes. MCC ranges from 0.68 to 0.70, further confirming the models' balanced classification ability. Their computational demands are modest, with sampling times around 2.3–2.4 s and training durations of approximately 70 s. These results suggest that synthetic oversampling is effective in alleviating the effects of class imbalance while preserving predictive accuracy, making it a reliable choice for financial distress prediction tasks.

Hybrid resampling techniques offer additional advantages by refining minority class learning and reducing noise in the feature space. SMOTE-Tomek, which combines SMOTE with Tomek Links, yields more balanced performance (recall: 0.69, precision: 0.75, F1-score: 0.72, MCC: 0.69) while slightly increasing computation time (5.9 s sampling, 68.5 s training), which is still acceptable for most financial applications. SMOTE-ENN achieves the highest recall among all oversampling variants (0.77) because its ENN component removes ambiguous or misclassified majority class instances, reducing the majority class to 13,702 samples (see Table 4), while its SMOTE component increases the minority class to 16,018 samples (see Table 4) by generating synthetic examples. This dual process enhances the model's ability to detect minority class cases. However, it may also introduce noise and eliminate some informative majority class samples, leading to more false positives and thus lower precision (0.63). As a result, the F1-score is moderate (0.69), reflecting the trade-off between improved recall and reduced precision. Its sampling and training times are 6.1 s and 58.1 s, respectively, indicating high computational efficiency.

Meanwhile, Bagging-SMOTE demonstrates competitive results across all metrics, achieving balanced recall (0.69), precision (0.74), and F1-score (0.72), supported by AUC (0.96), PR-AUC (0.80), and MCC (0.68). Although its training time is relatively long (400.3 s), the combination of oversampling and ensemble learning significantly enhances generalization performance and stability, especially for highly imbalanced datasets such as those used in financial distress prediction.

Table 6 compares the performance of these resampling techniques on the test set using confusion matrix metrics: True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP). These outcomes are further illustrated in Figure 6. The results highlight the trade-offs inherent in each method, particularly the balance between enhancing the detection of minority (i.e., TP: Class 1, distressed) cases and preserving the accuracy of majority (i.e., TN: Class 0, non-distressed) classifications.

Using the model trained without any resampling as a baseline, it achieved high accuracy in identifying non-distressed samples (TN = 6859; FP = 102), reflecting a strong bias toward the majority class. However, its performance in detecting distressed cases was notably weaker (TP = 575; FN = 379), highlighting the adverse impact of class imbalance. The resulting low sensitivity undermines the model's practical effectiveness in early financial distress detection, where accurate identification of minority cases is critical.

To mitigate this issue, we compared a series of resampling techniques, including oversampling, undersampling, and hybrid techniques. Among the oversampling methods, SMOTE, ADASYN, and Borderline-SMOTE all balanced the training set to a 1:1 ratio and improved recall for the minority class. SMOTE achieved TP = 661 (FN = 293), ADASYN yielded TP = 653 (FN = 301), and Borderline-SMOTE produced TP = 659 (FN = 295). However, these improvements in sensitivity came at the cost of increased false positives, 198,

210, and 202, respectively, compared to only 102 in the baseline model without resampling. The results suggest that while these methods enhance minority detection, they do so at the cost of greater misclassification of non-distressed instances, thereby reducing overall precision.

Table 6. Confusion matrix on the test set.

Methods	Test Set Class 0: 6961, Class 1: 954			
	TN	**FP**	**FN**	**TP**
Original data	6859	102	379	575
SMOTE	6763	198	293	661
ADASYN	6751	210	301	653
Borderline-SMOTE	6759	202	295	659
RUS	6040	921	117	837
Tomek Links	6833	128	370	584
SMOTE-Tomek	6746	215	295	659
SMOTE-ENN	6528	433	217	737
Bagging-SMOTE	6734	227	291	663

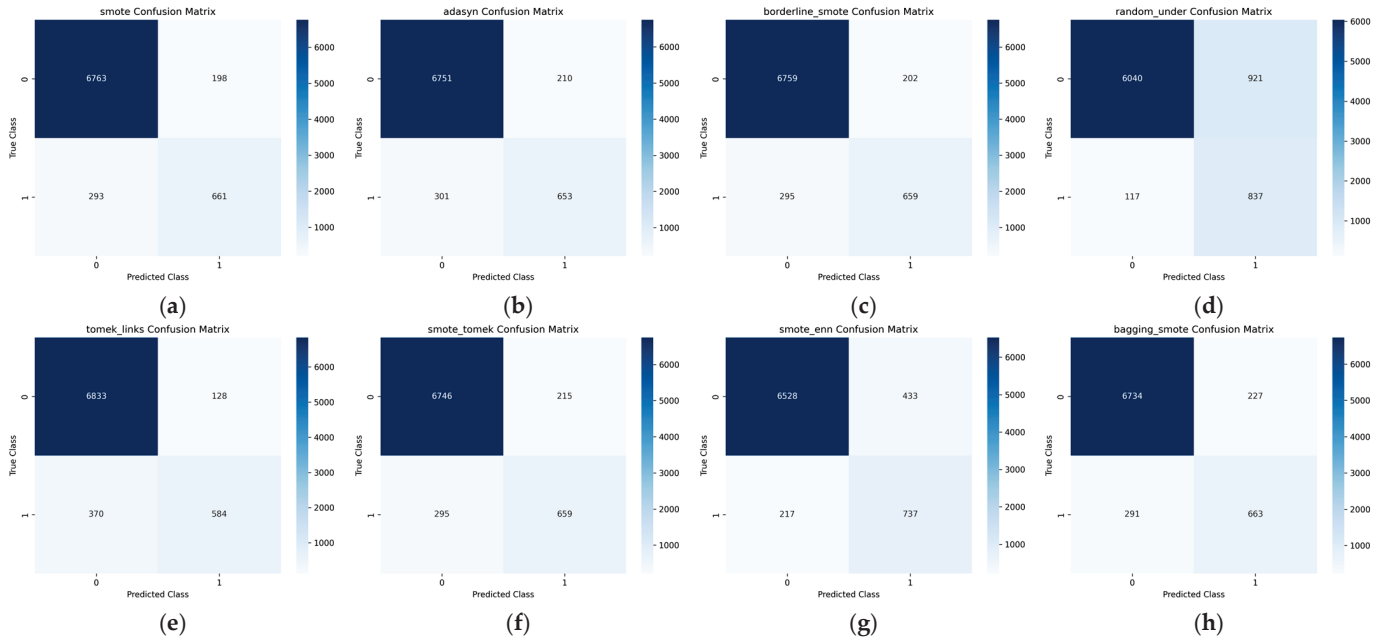


Figure 6. The comparison of the confusion matrix. (**a**) SMOTE; (**b**) ADASYN; (**c**) Borderline-SMOTE; (**d**) RUS; (**e**) Tomek Links; (**f**) SMOTE-Tomek; (**g**) SMOTE-ENN; (**h**) Bagging-SMOTE.

In contrast, RUS, which aggressively balanced the classes by reducing the majority to match the minority, achieved the highest recall (TP = 837; FN = 117). However, this came with the highest false positive count (FP = 921) and a reduction in true negatives to 6040, indicating substantial information loss and impaired model precision. A more conservative undersampling approach, Tomek Links, preserved most of the majority data, modestly improved recall (TP = 584), and maintained a relatively low FP (128). This confirms that selective undersampling can improve minority detection without drastically compromising specificity.

Hybrid methods, such as SMOTE-Tomek and SMOTE-ENN, aimed to combine the advantages of both oversampling and data cleaning. SMOTE-Tomek achieved a balanced training set and yielded TP = 659, FN = 295, and FP = 215, offering a better balance between

sensitivity and specificity compared to SMOTE alone. In contrast, SMOTE-ENN produced a higher TP (737) and lower FN (217) but incurred a sharp increase in FP (433), resulting in a significant drop in precision. While SMOTE-ENN demonstrated strong recall, its high false alarm rate makes it less desirable in practical scenarios requiring high precision.

Finally, Bagging-SMOTE demonstrated a strong balance between sensitivity and specificity. It achieved TP = 663, FN = 291, and FP = 227, resulting in a sensitivity of 69.5% and a specificity of 96.8%. Compared to standard SMOTE (TP = 661, FN = 293, FP = 198), Bagging-SMOTE delivered marginal improvements in both sensitivity and specificity, albeit with a slight increase in false positives. The ensemble design effectively stabilizes predictions by averaging across multiple base learners, thereby reducing the variance typically introduced by oversampling. These results suggest that Bagging-SMOTE enhances the model's generalization capability in the context of highly imbalanced financial distress prediction tasks.

5. Discussion

The experimental analysis highlights the crucial role of class imbalance handling in financial distress prediction. In the highly imbalanced dataset used, where distressed samples represent a small minority, XGBoost tended to bias predictions toward the majority (non-distressed) class, resulting in high precision but poor sensitivity to the minority class (such as the model with no sampling). This necessitates the adoption of resampling techniques to mitigate the imbalance and improve minority class detection.

The results revealed substantial performance variability across different resampling techniques, particularly in metrics sensitive to minority class detection, such as recall, precision, and F1-score. Standard SMOTE achieved the highest F1-score (0.73) and the best Matthews Correlation Coefficient (MCC) score (0.70) among all tested techniques. These results indicated that SMOTE was particularly effective in improving the model's ability to detect distressed samples while maintaining a sound balance between precision and recall. The computational demands of synthetic oversampling methods (such as SMOTE and ADASYN) were also modest, with sampling times around 2.3–2.4 s and training durations of approximately 70 s, making them practical for real-world applications. The synthetic generation of minority class samples through interpolation appeared to have successfully generalized the minority class distribution in the feature space, thereby enhancing classifier robustness without introducing significant noise or overfitting.

Among the oversampling and hybrid techniques, Borderline-SMOTE and SMOTE-Tomek consistently delivered more balanced outcomes. For example, compared to standard SMOTE, both techniques achieved similar or slightly improved recall (up to 0.69 for SMOTE-Tomek), highlighting their enhanced ability to identify distressed samples. This suggests that targeted oversampling near decision boundaries (as in Borderline-SMOTE) and the hybrid removal of overlapping instances (as in SMOTE-Tomek) can improve the model's sensitivity to "hard-to-classify" cases while preserving the integrity of the majority class distribution. SMOTE-Tomek offered a good trade-off between recall (0.69), precision (0.75), and computational efficiency (sampling: 5.9 s, training: 68.5 s). These findings reinforce the value of SMOTE as a strong baseline oversampling technique in high-stakes financial prediction tasks while also emphasizing the added benefits of more advanced resampling techniques that focus on decision boundary refinement and noise reduction.

In contrast, Random Undersampling (RUS) achieved the highest recall (0.88) but at the cost of much lower precision (0.48) and overall accuracy (0.87). However, RUS was the most computationally efficient among all methods (training time: 14.9 s, sampling time: 1.6 s), making it suitable for scenarios where detection sensitivity is prioritized and resources are limited. Tomek Links also offered a time-efficient option (sampling: 2.7 s,

training: 41 s) with strong precision (0.82) and moderate recall (0.61), making it attractive for applications where computational speed is critical.

In the Bagging-SMOTE technique, stratified SMOTE was independently applied to each bootstrapped subset of the training data using a sampling ratio of 0.15, and final predictions were obtained through ensemble averaging. While Bagging-SMOTE did not achieve the highest performance in all evaluation metrics, it demonstrated comparable and stable results, including an AUC of 0.96, F1-score of 0.72, and PR-AUC of 0.80. Notably, Bagging-SMOTE maintained a balanced trade-off between precision (0.74) and recall (0.69) and yielded a stable MCC of 0.68, indicating strong generalization capability. However, this came at the cost of a much longer training time (400.3 s), which may limit its practicality in time-sensitive applications. Its performance was particularly notable given the minimal modification of the original dataset, which preserved data integrity while still improving minority class detection. These results underscore the practical effectiveness of combining stratified oversampling at an appropriate sampling ratio with ensemble learning, particularly in enhancing the detection of minority-class (distressed) cases while mitigating model variance, a trade-off that is especially valuable in real-world financial scenarios where both predictive accuracy and data fidelity are critical.

In real-world financial distress prediction, the costs associated with false positives (incorrectly flagging a healthy company as distressed) and false negatives (failing to identify an actually distressed company) are highly asymmetric and context dependent. False negatives are often more detrimental, as overlooking a distressed firm can lead to significant financial losses, regulatory penalties, or reputational harm. In contrast, false positives may result in unnecessary interventions, increased monitoring costs, or missed business opportunities. Therefore, the choice of resampling technique should go beyond overall accuracy and consider the specific cost structure, business objectives, and risk tolerance of the application.

For example, techniques that emphasize recall, such as Borderline-SMOTE or SMOTE-Tomek, may be preferred in early warning systems, where minimizing missed distress cases outweighs the cost of false alarms. Conversely, when operational efficiency and resource management are paramount, methods such as Bagging-SMOTE, which provide a more balanced trade-off between precision and recall, may be more appropriate. Additionally, when computational resources or time constraints are a concern, RUS or Tomek Links may be prioritized due to their superior efficiency, despite their trade-offs in predictive performance.

Overall, the experimental findings highlight the importance of selecting resampling techniques that are closely aligned with the specific objectives and misclassification costs of the application to maximize the real-world value of financial distress prediction models.

Future research may extend this work by investigating uncertainty-aware sampling frameworks, where sample selection is guided by measures of predictive confidence or entropy. Additionally, incorporating explainable machine learning techniques, such as SHAP, to assess the influence of synthetic samples on model behavior could provide valuable insights for refining the resampling-model feedback loop. These advancements hold potential to support the development of more intelligent, interpretable, and effective resampling strategies, particularly critical in high-stakes domains such as financial risk assessment and corporate distress prediction.

6. Conclusions

In this study, we systematically compared eight resampling techniques to address the class imbalance problem in financial distress prediction, using a real-world dataset comprising 26,383 firm-quarter samples from 639 Chinese A-share listed companies spanning

the years 2007 to 2024. With only 12.1% of the samples labeled as distressed, the dataset presents a typical yet challenging imbalanced classification task. To ensure consistency and fair evaluation, the XGBoost algorithm was employed as the base classifier across all techniques, with identical model structures and hyperparameter settings.

The experimental results show that resampling techniques have a significant impact on model performance. This effect is especially notable in metrics sensitive to minority-class detection, such as precision, recall, F1-score, and Matthews Correlation Coefficient (MCC). Several key findings emerged from the comparative analysis:

- Resampling is critical for improving minority class detection. Traditional oversampling methods, especially SMOTE, showed meaningful improvements in F1-score (up to 0.73) and MCC (up to 0.70). These results indicate enhanced model robustness without excessive overfitting. Hybrid techniques such as SMOTE-Tomek and Borderline-SMOTE further improved recall while almost maintaining precision. This highlights their value in refining decision boundaries and reducing class overlap. Importantly, these methods also demonstrated modest computational demands, with sampling times around 2–6 s and training times of approximately 70 s. Therefore, they are practical for real-world applications.
- Trade-offs between recall and precision, as well as computational efficiency, must be context aware. Aggressive techniques such as RUS yielded the highest recall (0.85). However, this came at the cost of poor precision (0.46) and diminished generalization performance, mainly due to the loss of informative majority-class samples. RUS was also the most computationally efficient method, with a training time of 14.9 s and a sampling time of 1.6 s. This makes it suitable for scenarios where detection sensitivity and speed are prioritized. In contrast, Tomek Links offered a strong balance between precision (0.82) and efficiency (sampling: 2.7 s, training: 41 s). This makes it attractive for applications where computational speed and minimizing false positives are critical. SMOTE-ENN achieved a high recall (0.77) but at a moderate precision level (0.63), suggesting a tendency toward increased false positives. Its computational cost was also reasonable (sampling: 6.1 s, training: 58.1 s).
- Bagging combined with stratified SMOTE (Bagging-SMOTE) achieved a strong overall balance across evaluation metrics. Applying stratified SMOTE with an appropriate sampling ratio (0.15) to each bootstrap sample and aggregating predictions through bagging led to competitive results. These included AUC of 0.96, F1-score of 0.72, PR-AUC of 0.80, and MCC of 0.68. However, this method required much longer training time (400.3 s), which may limit its practicality in time-sensitive applications. These results show that ensemble-based resampling can improve both sensitivity and generalization, which is important in financial applications.
- Resampling techniques selection should be tailored to application goals. For early warning systems that prioritize recall and timely identification of financially distressed companies, techniques such as Bagging-SMOTE, Borderline-SMOTE, SMOTE-Tomek, and SMOTE are recommended. In contrast, when minimizing false positives is important, such as in regulatory flagging or risk-adjusted decision-making, more conservative methods, such as Tomek Links, may be more appropriate.

Practitioners should carefully consider the specific cost structure and operational priorities of their financial distress prediction tasks. For applications where missing a distressed firm is particularly costly, we recommend prioritizing resampling methods that maximize recall, such as Bagging-SMOTE or Borderline-SMOTE. Conversely, in scenarios where false alarms are more problematic, more conservative techniques such as Tomek Links may be preferable. Additionally, when computational resources or time constraints are a concern, RUS and Tomek Links offer practical alternatives due to their superior

efficiency, despite their trade-offs in predictive performance. Ensemble-based approaches can offer a robust balance between sensitivity and specificity, making them suitable for high-stakes financial environments.

Future research could advance these findings by integrating uncertainty-aware sampling strategies, where sample selection is driven by predictive confidence. Leveraging explainable machine learning techniques could further improve the synergy between resampling and modeling, while enhancing the transparency of interpretability. Additionally, extending these techniques to datasets from diverse markets, such as the U.S. or Europe, would provide valuable insights into their generalizability, making a valuable avenue for future exploration. Researchers could also investigate advanced class imbalance handling methods, such as adaptive resampling or hybrid approaches, combined with cost-sensitive learning. Finally, real-world case studies are needed to evaluate the practical impact and deployment of these techniques in financial risk assessment and early warning systems.

Author Contributions: Conceptualization, G.H., D.L.T., S.Y.L. and P.Y.C.; Formal analysis, G.H.; Investigation, G.H.; Methodology, G.H. and D.L.T.; Writing—original draft, G.H. and D.L.T.; Writing—review & editing, D.L.T., S.Y.L. and P.Y.C.; Validation, D.L.T., S.Y.L. and P.Y.C.; Visualization, G.H., Supervision, D.L.T., S.Y.L. and P.Y.C.; Project administration, S.Y.L.; Funding acquisition, D.L.T., S.Y.L. and P.Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported with funding from UTARRF, project number IPSR/RMC/UTARRF/2024-C1/T07.

Data Availability Statement: The original contribution presented in the research is included in the paper. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors are grateful to the anonymous referees for their comments, which substantially improved the quality of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Song, Y.; Jiang, M.; Li, S.; Zhao, S. Class-imbalanced Financial Distress Prediction with Machine Learning: Incorporating Financial, Management, Textual, and Social Responsibility Features into Index System. *J. Forecast.* **2024**, *43*, 593–614. [CrossRef]
2. Engïn, U. Financial distress prediction from time series data using xgboost: Bist100 of borsa istanbul. *Doğuş Üniversitesi Derg.* **2023**, *24*, 589–604. [CrossRef]
3. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
4. Wang, B.; Mei, C.; Wang, Y.; Zhou, Y.; Cheng, M.-T.; Zheng, C.-H.; Wang, L.; Zhang, J.; Chen, P.; Xiong, Y. Imbalance Data Processing Strategy for Protein Interaction Sites Prediction. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **2021**, *18*, 985–994. [CrossRef]
5. Cai, T. Breast Cancer Diagnosis Using Imbalanced Learning and Ensemble Method. *Appl. Comput. Math.* **2018**, *7*, 146. [CrossRef]
6. Elreedy, D.; Atiya, A.F. A Novel Distribution Analysis for SMOTE Oversampling Method in Handling Class Imbalance. In *Computational Science–Proceedings of the ICCS 2019: 19th International Conference, Faro, Portugal, 12–14 June 2019*; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2019; pp. 236–248. [CrossRef]
7. Alex, S.A.; Nayahi, J.J.V. Classification of Imbalanced Data Using SMOTE and AutoEncoder Based Deep Convolutional Neural Network. *Int. J. Unc. Fuzz. Knowl. Based Syst.* **2023**, *31*, 437–469. [CrossRef]
8. Han, H.; Wang, W.-Y.; Mao, B.-H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Advances in Intelligent Computing*; Springer Berlin Heidelberg: Berlin, Heidelberg, 2005; Volume 3644, pp. 878–887. ISBN 978-3-540-28226-6. [CrossRef]
9. Chen, C.; Shen, W.; Yang, C.; Fan, W.; Liu, X.; Li, Y. A New Safe-Level Enabled Borderline-SMOTE for Condition Recognition of Imbalanced Dataset. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–10. [CrossRef]
10. Glazkova, A. A Comparison of Synthetic Oversampling Methods for Multi-Class Text Classification. *arXiv* **2020**, arXiv:2008.04636. [CrossRef]

11. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; IEEE: Hong Kong, China, 2008; pp. 1322–1328. [CrossRef]
12. Pristyanto, Y.; Nugraha, A.F.; Dahlan, A.; Wirasakti, L.A.; Ahmad Zein, A.; Pratama, I. Multiclass Imbalanced Handling Using ADASYN Oversampling and Stacking Algorithm. In Proceedings of the 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Republic of Korea, 3 January 2022; IEEE: Seoul, Republic of Korea, 2022; pp. 1–5. [CrossRef]
13. Arifin, M.A.S.; Stiawan, D.; Yudho Suprapto, B.; Susanto, S.; Salim, T.; Idris, M.Y.; Budiarto, R. Oversampling and Undersampling for Intrusion Detection System in the Supervisory Control and Data Acquisition IEC 60870-5-104. *IET Cyber-Phys. Syst. Theory Appl.* **2024**, *9*, 282–292. [CrossRef]
14. Batista, G.E.A.P.A.; Prati, R.C.; Monard, M.C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explor. Newsl.* **2004**, *6*, 20–29. [CrossRef]
15. Kaushik, M.M.; Mahmud, S.M.H.; Kabir, M.A.; Nandi, D. The Effects of Class Rebalancing Techniques on Ensemble Classifiers on Credit Card Fraud Detection: An Empirical Study. In Proceedings of the Applied Data Science and Smart Systems, Rajpura, India, 4–5 November 2022; AIP Publishing LLC: Rajpura, India, 2023; p. 030011. [CrossRef]
16. Zang, J.; Li, H. Abnormal Traffic Detection Based on Data Augmentation and Hybrid Neural Network. In Proceedings of the 2024 2nd International Conference on Signal Processing and Intelligent Computing (SPIC), Guangzhou, China, 20 September 2024; IEEE: Guangzhou, China, 2024; pp. 249–253. [CrossRef]
17. Putra, L.G.R.; Marzuki, K.; Hairani, H. Correlation-Based Feature Selection and Smote-Tomek Link to Improve the Performance of Machine Learning Methods on Cancer Disease Prediction. *Eng. Appl. Sci. Res.* **2023**, *50*, 577583. [CrossRef]
18. Yang, F.; Wang, K.; Sun, L.; Zhai, M.; Song, J.; Wang, H. A Hybrid Sampling Algorithm Combining Synthetic Minority Over-Sampling Technique and Edited Nearest Neighbor for Missed Abortion Diagnosis. *BMC Med. Inform. Decis. Mak.* **2022**, *22*, 344. [CrossRef]
19. Wang, W.; Liang, Z. Financial Distress Early Warning for Chinese Enterprises from a Systemic Risk Perspective: Based on the Adaptive Weighted XGBoost-Bagging Model. *Systems* **2024**, *12*, 65. [CrossRef]
20. Liu, W.; Fan, H.; Xia, M.; Pang, C. Predicting and Interpreting Financial Distress Using a Weighted Boosted Tree-Based Tree. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105466. [CrossRef]
21. Wu, C.; Chen, X.; Jiang, Y. Financial Distress Prediction Based on Ensemble Feature Selection and Improved Stacking Algorithm. *Kybernetes*, 2024; *ahead-of-print*. [CrossRef]
22. Seiffert, C.; Khoshgoftaar, T.M.; Van Hulse, J.; Napolitano, A. RUSBoost: A Hybrid Approach to Alleviating Class Imbalance. *IEEE Trans. Syst. Man Cybern. A* **2010**, *40*, 185–197. [CrossRef]
23. Díez López, C.; Montiel González, D.; Vidaki, A.; Kayser, M. Prediction of Smoking Habits From Class-Imbalanced Saliva Microbiome Data Using Data Augmentation and Machine Learning. *Front. Microbiol.* **2022**, *13*, 886201. [CrossRef]
24. Shaikh, S.; Daudpota, S.M.; Imran, A.S.; Kastrati, Z. Towards Improved Classification Accuracy on Highly Imbalanced Text Dataset Using Deep Neural Language Models. *Appl. Sci.* **2021**, *11*, 869. [CrossRef]
25. Kotb, M.H.; Ming, R. Comparing SMOTE Family Techniques in Predicting Insurance Premium Defaulting Using Machine Learning Models. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 621–629. [CrossRef]
26. Hairani, H.; Anggrawan, A.; Priyanto, D. Improvement Performance of the Random Forest Method on Unbalanced Diabetes Data Classification Using Smote-Tomek Link. *JOIV Int. J. Inform. Vis.* **2023**, *7*, 258. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

mathematics

MDPI

Article

A Generative Adversarial Network-Based Investor Sentiment Indicator: Superior Predictability for the Stock Market

Shiqing Qiu [1], Yang Wang [2], Zong Ke [3,*], Qinyan Shen [4], Zichao Li [5], Rong Zhang [6] and Kaichen Ouyang [7,*]

[1] School of Mathematical Sciences, Chengdu University of Technology, Chengdu 610059, China; qiusq0410@163.com
[2] Department of Mathematics, University of Chicago, Chicago, IL 60637, USA; wangyang@uchicago.edu
[3] Faculty of Science, National University of Singapore, Singapore 119077, Singapore
[4] Department of Statistics, University of South Carolina, Columbia, SC 29201, USA; qshen@email.sc.edu
[5] The Department of Management Science and Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada; zichao.li@uwaterloo.ca
[6] Graduate School of Management, University of California, Davis, CA 95616, USA; rzhang1118@gmail.com
[7] Department of Mathematics, University of Science and Technology of China, Hefei 230026, China
* Correspondence: a0129009@u.nus.edu (Z.K.); oykc@mail.ustc.edu.cn (K.O.)

Abstract: Investor sentiment has a profound impact on financial market volatility; however, it is difficult to accurately capture the complex nonlinear relationships among sentiment proxies with the existing methods. In this study, we propose a novel investor sentiment indicator, S^{GAN}, which uses generative adversarial networks (GANs) to extract the nonlinear latent structure from eight sentiment proxies from February 2003 to September 2023 in the Chinese A-share market. Unlike traditional linear dimensionality reduction methods, GANs are able to capture complex market dynamics through adversarial training, effectively reducing noise and improving prediction accuracy. The empirical analyses show that S^{GAN} significantly outperforms existing methods in both in-sample and out-of-sample prediction capabilities. The GAN-based investment strategy achieves impressive annualized returns and provides a powerful tool for portfolio construction and risk management. Robustness tests across economic cycles, industries, and U.S. markets further validate the stability of S^{GAN}. These findings highlight the unique advantages of GANs as sentiment-driven financial forecasting tools, providing market participants with new ways to more accurately capture sentiment-shifting trends and develop effective investment strategies.

Keywords: investor sentiment; generative adversarial networks (GANs); dimensionality reduction; return predictability; financial mathematics

MSC: 68T07

1. Introduction

Optimistic investors typically respond positively to market fluctuations, whereas pessimistic investors exhibit adverse reactions [1,2]. Extensive studies have confirmed the significant impact of investor sentiment on capital market fluctuations [3,4]. A comprehensive understanding of investor sentiment is essential for predicting stock market returns and formulating effective investment strategies [5].

Early research on investor sentiment frequently relied on survey-based measurements [6]. For example, in the paper by Hao, the consumer confidence index (CCI) was introduced to determine sentiment levels on specific dates [7]. Dong et al. used text mining

on daily Twitter feeds, employing neural networks and two analysis tools for sentiments [8]. Some researchers employed principal component analysis (PCA) to formulate a synthetic sentiment gauge [9]. While the choice of sentiment proxies can vary, PCA remains widely used in sentiment analysis [10–12]. The partial least squares (PLS) method improves predictive accuracy by filtering out irrelevant noise [13]. Later, Huang et al. introduced the sPCA method for predicting Chinese stock market returns [14]. Recently, the genetic algorithm (GA), which optimizes complex problems by simulating biological evolution, has also been increasingly applied to dimensionality reduction tasks [15].

Beyond PCA, sPCA, PLS, and GA, numerous other dimensionality reduction methods exist. Since their inception, generative adversarial networks (GANs) have revolutionized artificial intelligence applications through their innovative dual-network architecture, particularly transforming progress in visual data analysis and linguistic pattern recognition [16]. The research of Kumar et al. implemented generative adversarial networks within a methodological framework designed to elevate the reliability of stock market forecasts and minimize errors in predictive outcomes [17]. The paper by Nejad and Ebadzadeh proposed an innovative model integrating GANs with feature-matching techniques to predict equity prices, demonstrating enhanced stability during training tests and mitigating mode collapse challenges in generative processes [18]. Wu et al. developed GAN-driven architectures incorporating the piecewise linear representation methodology to model three distinct trading strategies in financial markets [19]. Xu et al. designed a self-regulated GAN framework through a methodological integration of adversarial and cooperative neural architectures to forecast fluctuations in equity prices, aiming to enhance predictive robustness in financial modeling [20]. The research by Yan and Li introduced the SAR-GAN, which integrates a self-attention mechanism with a residual network to forecast leading stocks in prominent industries across various markets [21]. So, in our paper, we investigate sentiment indicators. In the investor sentiment literature, the term 'sentiment' is often denoted by S, and the specific method of generation is labelled on it. For example, S^{GAN} denotes the investor sentiment indicator generated by a GAN, S^{GA} denotes the investor sentiment indicator generated by a genetic algorithm, S^{PLS} denotes indicators generated through partial least squares, and S^{PCA} denotes indicators generated through principal component analysis. This notation is generally accepted and followed by researchers as the standard way of writing [14].

Our study specifically aims to address three key research questions. Firstly, can GANs create a more effective investor sentiment indicator compared to traditional dimensionality reduction methods? Moreover, does our proposed S^{GAN} demonstrate superior predictive power for stock market returns in both in-sample and out-of-sample tests? Finally, how robust is the predictive performance of S^{GAN} across different market conditions, economic cycles, and industries?

This study advances prior scholarly discourse and application through the following key dimensions. Firstly, we apply GANs to the field of investor sentiment indicator construction, which is fundamentally different from linear dimensionality reduction methods such as PCA and PLS in the existing literature. GANs are capable of capturing complex nonlinear relationships among sentiment agents through the adversarial learning mechanism of generators and discriminators, a capability that is particularly important in the highly nonlinear financial markets. The existing literature on financial sentiment analysis rarely explores the potential of adversarial neural networks for sentiment quantification; thus, this study fills this methodological gap. Secondly, by comparing with the investor sentiment indicators constructed using the existing methodology, it is verified that the newly constructed sentiment indicator performs better in market forecasting. Moreover, out-of-sample experiments, multi-period analysis, multi-factor testing, and analysis of

different industries, periods, and countries are used to demonstrate the strong robustness of S^{GAN}. Last but not least, the findings of our paper enable investors to grasp the tendency of the change in investor sentiments better, which could help to construct more effective investing strategies.

This paper is structured as follows: Section 2 provides a summary of the data, while Section 3 outlines the methodology employed. Section 4 conducts empirical analyses, followed by a robustness check in Section 5. Finally, Section 6 concludes this study.

2. Data

2.1. Data Sources and Variables

We carefully collected data from CSMAR and the EPU website, selecting the data from February 2003 to September 2023. In this paper, the monthly frequency return of Shanghai, Shenzhen, and Beijing A-share markets is defined as the Chinese stock market returns. Excess market returns (R^m) are computed by deducting the risk-free rate from the total returns generated by the A-share market index. We also collect the PCA-based investor sentiment S^{PCA} from CSMAR from February 2003 to September 2023.

This paper examined eight well-established proxies closely associated with sentiment trends. These proxies include fund discount rates [22,23], trading volume in the last month [24], IPO frequency [9], first-day IPO returns [9], monthly growth in investor accounts [9], consumer confidence index [25], equity return volatility [26], and the economic policy uncertainty index [27–29]. Below, Table 1 provides a concise overview of these eight sentiment indicators.

Table 1. The eight sentiment indicators and their definitions.

Variable	Definition
Fund discount rates (*DCEF*)	The comprehensive discount rate of market funds weighted by fund shares.
Trading volume in last month (*TURN*)	The rate of monthly trading volume relative to total market.
IPO frequency (*IPON*)	The count of IPOs within that month.
First-day IPO returns (*IPOR*)	The average returns of the IPO.
Monthly growth in investor accounts (*NIA*)	The logarithm of the monthly new investor registrations.
Consumer confidence index (*CCI*)	Constructed using data from quarterly surveys administered by CEMA.
Equity return volatility (*RV*)	Monthly return volatility is estimated through the summation of squared daily returns.
Economic policy uncertainty index (*EPU*)	The natural logarithmic of the EPU index.

Notes: This table provides the eight sentiment indicators and their definitions. The eight indicators are fund discount rates, trading volume in last month, IPO frequency, first-day IPO returns, monthly growth in investor accounts, consumer confidence index, equity return volatility, and economic policy uncertainty index. The period of the sample is from February 2003 to September 2023.

Table 2 shows the summary of the variables of the paper, which include the excess market returns (R^m), the three different investment sentiments developed by different methods, and the eight sentiment proxies [30]. Figures 1 and 2 show the excess return and S^{GAN}, respectively. From the two figures, we notice that the trends of Rm and S^{GAN} are fairly similar. For example, from December 2007 to July 2009, there was a significant and sudden drop in both R^m and S^{GAN}, which could be attributed to the world financial crisis triggered by the 2008 subprime mortgage crisis in the United States. At the same time, this implies that S^{GAN} may be a better predictor of excess returns.

Table 2. Summary statistics.

Variables	Mean	Max	Min	Std	Skew	Kurt	Obs
R^m	0.008	0.296	−0.268	0.078	−0.062	1.768	248
S^{GAN}	0.008	0.161	−0.1712	0.040	−0.392	4.012	248
S^{PLS}	0.000	2.316	−2.214	0.794	−0.003	0.314	248
S^{GA}	0.008	0.077	−0.092	0.028	−0.538	1.018	248
DCEF	−8.084	0.984	−37.501	11.133	−1.561	0.953	248
TURN	0.243	0.689	0.080	0.117	1.447	2.191	248
IPON	17.383	82.000	0.000	15.844	0.966	0.658	248
IPOR	5.786	39.281	−4.356	6.966	2.476	7.513	248
NIA	13.564	16.000	11.140	1.028	−0.587	0.043	248
CCI	103.924	127.000	85.500	11.278	0.457	−0.840	248
RV	0.014	0.045	0.004	0.007	1.688	3.153	248
EPU	161.822	661.828	23.716	120.769	1.407	1.806	248

(Notes: The statistics of this table include mean (Mean), maximum (Max), minimum (Min), standard deviation (Std), skewness (Skew), kurtosis (Kurt), and observations (Obs). The period of this sample is from February 2003 to September 2023.)

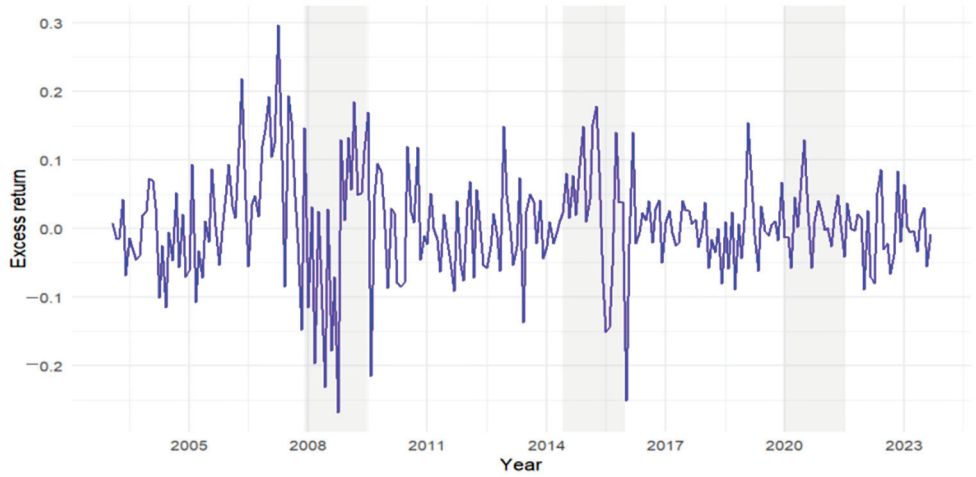


Figure 1. The excess return in the Chinese A-share stock market.

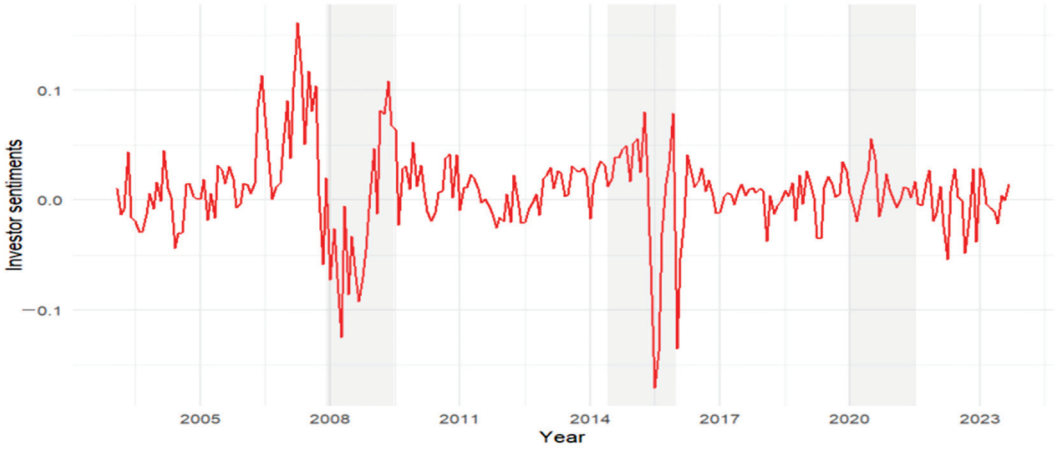


Figure 2. The dynamic of S^{GAN}.

2.2. Data Processing

Out of a total of 248 monthly samples, we found missing values for about 4.2 per cent of the data points, which were mainly concentrated on the indicators DCEF (seven missing points), CCI (three missing points), and EPU (one missing point). For these missing data, we use linear interpolation to populate them in order to maintain the continuity

of the time series. Our sampling process is based on a full sample collection covering all available monthly data from February 2003 to September 2023, and this continuous time series sampling methodology is able to comprehensively capture sentiment changes across market cycles. For outlier handling, we use three times the standard deviation (3σ) as the threshold, identifying and replacing about 2.3% of the outlier data points to ensure that model training is not disturbed by extreme values [16]. Smooth data in our study refers to data where the statistical properties of the time series remain constant over time. We use the ADF test to assess data smoothness. The original hypothesis was that the series is nonstationary if it contains a unit root. The results of the test showed that some of the original sentiment proxy variables were nonstationary (p-value > 0.05); so, we used first-order differencing to transform these series into stationary series. Although the difference treatment changes the absolute value of the data, it retains the relative change information of the series, which is particularly important for capturing the dynamics of investor sentiment. Finally, to address disparities in measurement scales across predictor variables, we standardized all the features, using the Z-score method, and the standardized data help to improve the stability and convergence speed of model training.

Furthermore, we examined potential multicollinearity among the eight sentiment proxies using VIF analysis. The VIF values ranged from 1.24 to 2.87, all well below the commonly accepted threshold of 10, indicating that multicollinearity is not a significant concern in our dataset. A correlation analysis further confirmed moderate correlations between the proxies (average correlation coefficient of 0.41), suggesting that each proxy contributes unique information to our sentiment indicator.

3. Methodology

3.1. The PCA Method (Principal Component Analysis)

Principal component analysis (PCA) first computes the covariance matrix of the data matrix and then derives its eigenvalues and corresponding eigenvectors [16]. This matrix contains the eigenvectors associated with the K features, where the eigenvector linked to the largest eigenvalue represents the direction of maximum variance. By projecting the original data onto this new space, the dimensionality of the dataset is reduced by PCA. The pseudocode for the PCA procedure is presented in Algorithm 1.

Algorithm 1. The pseudocode of principal component analysis.

Require: Data matrix X, target dimension k
Compute mean vector : $\mu \leftarrow \frac{1}{n}\sum_{i=1}^{n} X_i$
Center the data : $X \leftarrow X - \mu$
Compute covariance matrix : $C \leftarrow \frac{1}{n-1}X^T X$
For each eigenvalue λ_i and eigenvector v_i of C do
    Solve for eigenvalues and eigenvectors : $Cv_i = \lambda_i v_i$
end for
Sort eigenvectors by eigenvalues in descending order
Select the top k eigenvectors : $W \leftarrow [v_1, v_2, \ldots, v_k]$
Project data onto the new space : $Y \leftarrow X \cdot W$
Return Reduced data matrix $Y = 0$

3.2. The PLS Method (Partial Least Squares)

The partial least squares method aims to capture essential information from various sentiment proxies while eliminating irrelevant noise, and its construction involves two main steps.

The first step includes several regressions of time series. Each proxy $x_{i,t}$ performs the time series regression on the realized stock return R_t. This regression model captures how each proxy responds to the underlying investor sentiment S_t, which is inferred through its relationship with stock returns.

$$x_{i,t} = \pi_{i,0} + \pi_i R_t + u_{i,t-1}(t = 1, 2, \cdots, T) \tag{1}$$

Here, π_i represents the loading of $x_{i,t-1}$ on the S_{t-1}, as instrumented using R_t.

The second step includes several cross-sectional regressions. In this step, we perform a cross-sectional regression for each period t, using the sentiment proxies $x_{i,t}$ and the loadings π_i obtained from the first step to estimate the aligned sentiment index S_t^{PLS}.

$$x_{i,t} = c_t + S_t^{PLS}\hat{\pi}_i + v_{i,t}(i = 1, 2, \cdots, N) \tag{2}$$

In this equation, S_t^{PLS} represents the estimated aligned investor sentiment, which is optimized to predict excess returns based on the covariance between the sentiment proxies and the returns.

The key formula for the S^{PLS} estimate using full-sample information can be written as:

$$S^{PLS} = XJ_N X' J_T R(R' J_T X J_N X' J_T R)^{-1} R' J_T R \tag{3}$$

Here, X is the matrix of sentiment proxies; R is the vector of excess market returns; J_T and J_N are matrices that adjust for constants in the regressions; and S^{PLS} is the vector of investor sentiment. Algorithm 2 below presents the pseudocode of partial least squares.

Algorithm 2. Pseudocode of partial least squares.

Require: Sentiment proxies (S), stock returns (R)
1: **for** each sentiment proxy (S_i) **do**
2: Perform time series regression : $S_i = \beta_i R + \varepsilon_i$
3: Store loadings (β_i)
4: **end for**
5: for each time period (t) **do**
6: Perform cross-sectional regression : $I_t = \sum \beta_i S_{i,t} + \eta_t$
7: **end for**
8: **return** (I_t)

3.3. The GA Method (Genetic Algorithm)

The genetic algorithm (GA) was proposed by Holland and then summarized and refined as a class of simulated evolutionary algorithms by Dejong and Goldberg [15]. It is an algorithm that searches for optimal solutions by simulating the mechanisms of inheritance in nature and the theory of biological evolution. This paper uses this algorithm to reduce multiple sentiment proxies into a single investor sentiment. The pseudocode for the GA is presented in Algorithm 3.

Algorithm 3. Pseudocode of genetic algorithm.

Require: Sentiment proxy matrix (X), target variable (Y), population size (N), mutation probability (P_m), max iterations (T)
function initializepopulation (N, m)
    $P \leftarrow$ Randomly generate N individuals with m features
    return P
end function
function evaluatefitness (P, X, Y)
    for each individual (p) in (P) do
        $R^2[p] \leftarrow$ LinearRegression $(X[p]\, Y)$
    end for
    return (R^2)
end function
function genericalgorithm (X, Y, N, P_m, T)
    $P \leftarrow$ initializepopulation $(N, len(X[0]))$
    for $t = 1$ to T do
        $(R^2) \leftarrow$evaluatefitness (P, X, Y)
        $P_{elite} \leftarrow$Select top individuals based on (R^2)
        $P_{new} \leftarrow$Crossover (P_{elite})
        $P_{mutated} \leftarrow$Mutation $(P_{new},\ P_m)$
        $P \leftarrow P_{mutated}$
    end for
    return Best individual as (S^{GA})
end function

3.4. The Generative Adversarial Networks (GANs)

3.4.1. The Principles of GANs

The objective is for the generator to learn a latent sentiment factor that effectively captures investor sentiment, while the discriminator ensures its quality.

(1) Mechanism of Generators and Discriminator

We choose to use the three-layer fully connected neural network instead of a convolutional or recurrent network, mainly because the input data no longer retain obvious temporal or local spatial structure after preprocessing, and a fully connected network may capture global features more directly and achieve effective dimensionality reduction. In addition, the simple structure and fewer parameters of fully connected networks make them easier to train and debug, and their flexibility and robustness are more likely to meet the needs in this task of dimensionality reduction and extraction of investor sentiment factors.

This sentiment factor reflects market optimism or pessimism by integrating sentiment proxies, thereby influencing investors' market expectations. Through adversarial training, the generator optimizes the extracted factor to enhance its explanatory power for excess market returns.

(2) Selection of activation function

To prevent the issue of gradient vanishing that may arise during training, we use the LeakyReLU activation function, which maintains a small slope in the negative input interval, thus improving the stability of training. The final layer does not use an activation function to give continuity to the output Z. The input to the discriminator is the low-dimensional investor sentiment variable Z, and its objective is to distinguish between real and generated samples.

3.4.2. Selection of Hyper-Parameters

(1) Optimization approach

The specific hyper-parameters are determined by a combination of grid search and cross-validation. During the grid search, we systematically explore different combinations of key parameters: learning rate, hidden layer size, and batch size. The initial range and step size of each parameter are shown in Table 3. The initial values of the hyper-parameters in Table 3 are not randomly set, but refer to the typical parameter ranges recommended by the existing literature on GANs in the field of time series analysis [17,18]. After performing the grid search, we analyze the results to determine the optimal hyper-parameter values that produce the best model performance [18,19].

Table 3. The initial range and step values of hyper-parameters.

	Initial Range	**Step Values**
Learning rate	[0.00001, 0.001]	(0.00001, 0.0001, 0.001)
Hidden layer size	[32, 256]	(32, 64, 128, 256)
Batch size	[16, 128]	(16, 32, 64, 128)

(Notes: Table 3 represents the initial range and step values of hyper-parameters, which include learning rate, hidden layer size, and batch size, respectively.).

Cross-validation is employed throughout the process to assess each hyper-parameter configuration, with the final selection based exclusively on out-of-sample R^2 and mean squared error (MSE). In our approach, the best performance is explicitly defined as the configuration that maximizes R^2 while minimizing MSE on the validation sets. In our tests, 80% of the data is allocated for training, while the remaining 20% serves as the validation set to assess the model's performance on unseen data in real time during training [16,17]. Additionally, a convergence threshold is defined through the 'patience' parameter in the early stopping strategy. If the validation loss fails to show significant improvement over 100 consecutive epochs, the model is considered to have reached a state of convergence, and the training is stopped, which not only prevents the GANs from overfitting when training, but also saves the computational resources and ensures that the final model has a better generalization ability. Although the hyper-parameter tuning process is computationally intensive due to the grid search and extensive cross-validation, this strategy effectively balances model complexity and generalization ability, ensuring robust performance across different market conditions.

For the selection of optimizer, we choose the Adam optimizer, which is able to avoid the common oscillation phenomenon in traditional gradient descent methods. The optimization function of the Adam optimizer updates the model parameters according to the losses of the generator and the discriminator, with the learning rate fixed at 0.0001. Last but not least, to ensure the loss values converge completely, we set the number of training epochs to 1000. Table 4 represents the hyper-parameters.

Table 4. The hyper-parameters in GANs.

Hyper-Parameters	**Value**
Hidden layer of the generator	64 to 32 to 1
Hidden layer of the discriminator	32 to 16 to 1
Learning rate	0.0001
Training epochs	1000

Notes: This table represents the hyper-parameters and their values used in GANs.

(2) Sensitivity analysis of hyper-parameters

In order to comprehensively assess the impact of hyper-parameters on the performance of the generator and subsequent regression models, this study conducts sensitivity analyses for four dimensions: learning rate, epochs, batch size, and depth of the generator network. By training the generator under different hyper-parameter configurations and calculating its tuning in the regression, it is possible to observe how the variation in these hyper-parameters affects the model's explanatory power of returns.

Figure 3 represents the results of the sensitivity experiments for each hyper-parameter. From these four sensitivity analysis plots, it could be seen that different learning rates, number of training epochs, batch sizes, and number of generator layers significantly affect the adjusted R^2 of the regression model. The convergence of the model improves significantly as the learning rate gradually increases from very low, and the increasing number of training epochs generally enhances the model's ability to explain returns, although fluctuations may occur between iterations. In terms of batch size, medium batches tend to strike a better balance between stability and training efficiency. At the same time, the increasing number of layers of the generator network may better capture complex mapping relationships and thus improve the interpretation of the model, but it also entails higher training difficulty and a potential risk of overfitting. Overall, these results suggest that moderate and balanced hyper-parameter tuning (e.g., choosing the right learning rate, number of training epochs, batch size, and reasonable network depth) for the data and methods could significantly improve the generator's dimensionality reduction effect on sentiment agents.

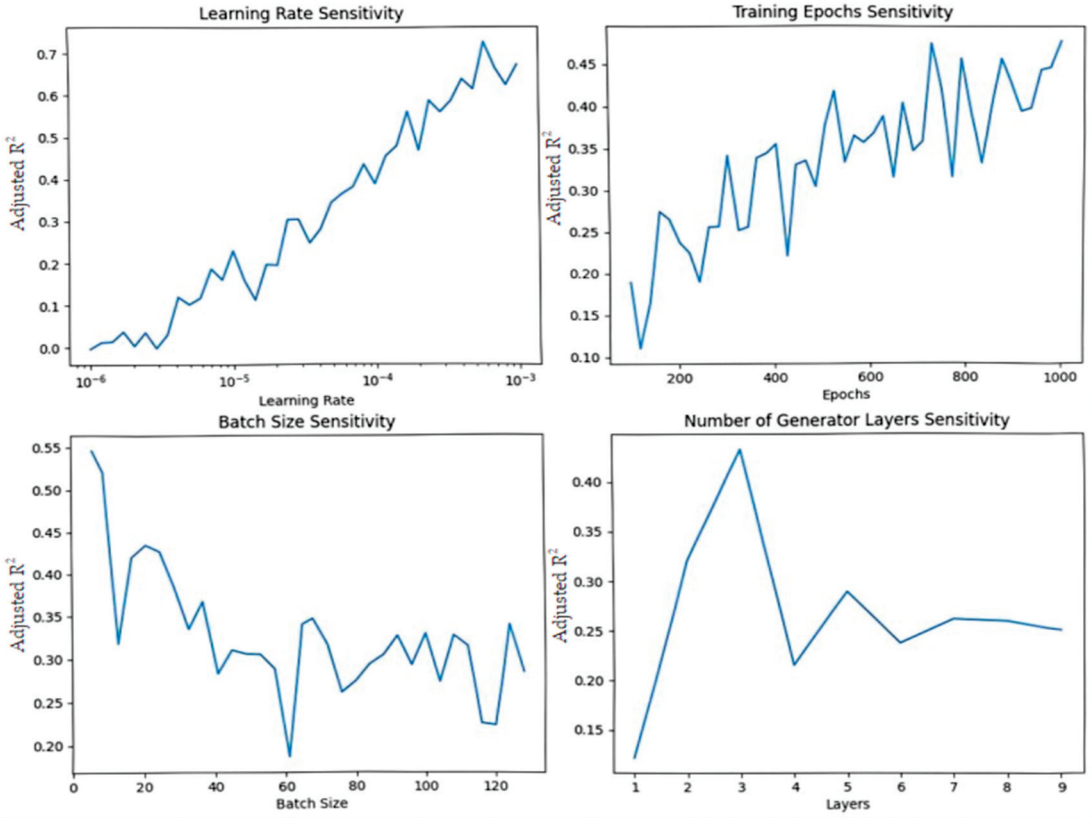


Figure 3. Sensitivity analysis of hyper-parameters.

3.4.3. Train Process of GANs

The details of this process are explained below, and Figure 4 represents the process diagram of the GANs.

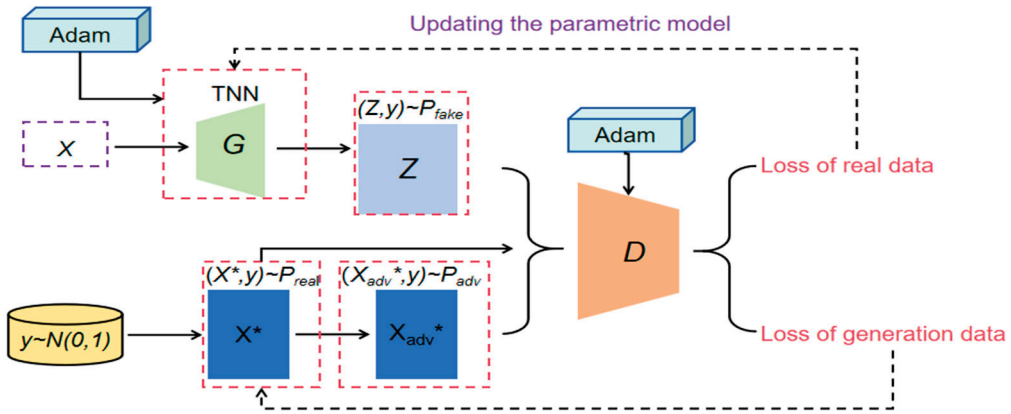


Figure 4. Flowchart of the GANs.

(1) Generator Training

① The generator takes high-dimensional financial sentiment proxies X as input, generating a low-dimensional sentiment factor.

② The loss function consists of adversarial loss (ensuring the generated sentiment factor z is indistinguishable from real data) and regression loss (MSE constraint) (maintaining predictive power for excess market returns). The function of adversarial loss is defined as:

$$L_{GAN} = E[\log D(z_{real})] + E[\log(1 - D(G(X)))] \tag{4}$$

$z_{real} - P_{real}(z)$ is the investor sentiment variables from the real market, and $z_{fake} = G(X)$ is the investor sentiments generated from X in the generator. Regression loss (MSE constraint) encourages the extracted sentiment factor z to maintain strong predictive power for excess market returns, which prevents mode collapse. MSE constraint is defined below.

$$L_{MSE} = E[(G(X) - y)^2] \tag{5}$$

In this equation, $G(X)$ is the investor sentiment factor output by the generator; y is excess market return.

③ The balance between adversarial and regression loss is fine-tuned through grid search. So, the total loss function of the generator is:

$$L_G = E[\log(1 - D(G(X)))] + \lambda E[(G(X) - y)^2] \tag{6}$$

In this equation, λ is for balancing adversarial loss and regression loss. Specifically, the value is chosen among multiple sets of parameters by the grid search method. We set up the grid search from 0 to 5 [31]. The experimental results show that $\lambda = 1$ can effectively prevent mode collapse while ensuring regression accuracy. The sensitivity analysis in Figure 5 shows that $\lambda = 1$ provides the least mean squared error; so, it verifies the reasonableness of this choice. We optimize the loss function using a gradient penalty technique. However, comparing the R^2 and MSE revealed that, due to the relative simplicity of the loss function employed in this network, this optimization strategy has only a marginal impact on the final results.

(2) Discriminator Training

① The discriminator distinguishes between real and generated sentiment factors.
② A binary classification loss function is used to minimize errors in differentiation.
③ Label smoothing (according to the research of Kumar, assigning real samples a label of 0.9 and fake samples 0.1) enhances training stability [21].

(3) Adversarial Optimization Process

① Fix G and train D to improve classification ability.
② Fix D and update G to generate more realistic sentiment factors.
③ The min–max game iterates until the generator learns a robust sentiment indicator.

(4) Maintaining Balance Between Generator and Discriminator

① To prevent mode collapse or instability, several strategies are implemented.
② Learning Rate Control: Both networks use Adam optimizer with a learning rate of 0.0001 for stable convergence.
③ Regularization Using MSE Constraint: Ensures extracted sentiment factors maintain economic interpretability.
④ Multiple Training Runs and Averaging: Training five independent GANs and averaging outputs mitigates random fluctuations.
⑤ Dropout Implementation: Prevents overfitting by randomly deactivating neurons during training.

(5) Avoidance of overfitting and mode collapse

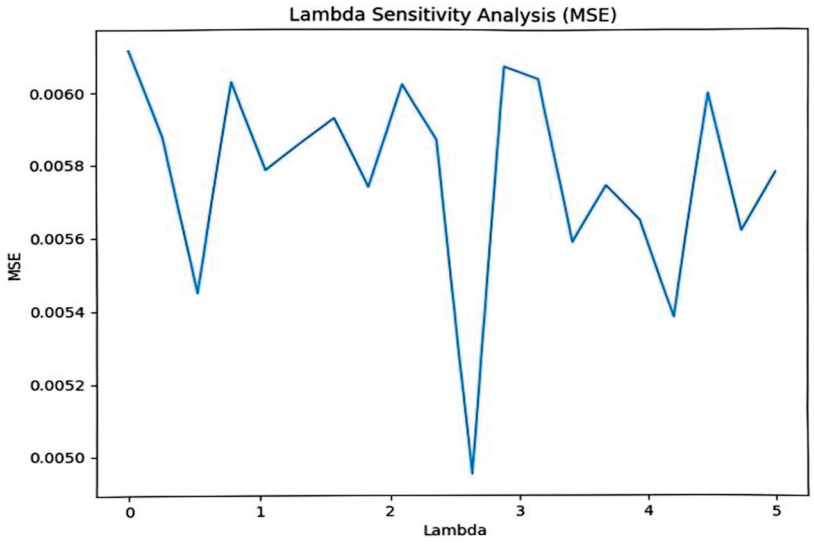


Figure 5. The sensitivity analysis of λ.

We mitigate pattern collapse by means of label smoothing. We use soft labels for the discriminator, setting the labels of the real samples to 0.9 instead of 1, and the labels of the fake samples to 0.1 instead of 0. This prevents the discriminator from becoming over-confident and thus provides the generator with smoother and informative gradient signals, which helps the generator to produce more diverse outputs. In addition, the code also increases training uncertainty during discriminator training by using random noise on the forged samples, which further promotes the generator to learn a richer feature distribution. Together, these two mechanisms help to mitigate the phenomenon of pattern collapse, ensuring that the generator is able to continuously improve and maintain the diversity of its output during adversarial training.

In order to prevent overfitting, the dropout mechanism is mainly used in this model. Dropout is applied in multiple hidden layers of the generator and discriminator, and its main function is to reduce the risk of overfitting by randomly discarding a portion of neurons and preventing the network from overdependence on some specific neurons [20]. By randomly masking a portion of the features at each forward propagation, dropout forces the model to learn more redundant and robust feature representations, which helps to improve the model's adaptability to data noise and variability. Additionally, the incorporation

of dropout enables the model to utilize the data more efficiently during training, avoiding the need to fit the training samples too finely, which in turn improves the generalization performance of the generator and the discriminator in the face of new data. This regularization method supports the stable training of the entire GAN framework and ensures that the generated investor sentiment can show good explanatory power in the regression model.

Figure 6 shows the learning curve of the training loss and validation loss, respectively. As shown in Figure 6, both the training and validation losses gradually decrease throughout the training process, staying closely aligned in the initial stages, which indicates that the model is effectively capturing the data's underlying patterns. As training advances, while the validation loss exhibits some fluctuations, it generally stabilizes, suggesting that the model maintains a strong fit for both the training and validation datasets. The validation loss does not continue to rise as it does in the overfitting case, but rather stabilizes at a lower level, indicating that the model is capable of generalizing effectively to the validation set while maintaining good training performance.

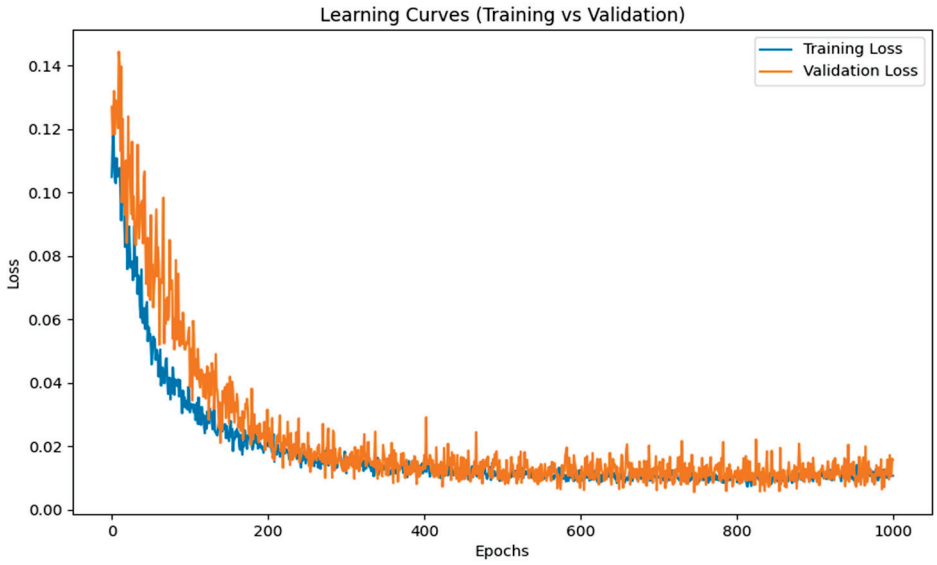


Figure 6. Learning curves of training loss and validation loss.

(6) Extracting the Sentiment Indicator

① The final investor sentiment indicator is extracted from the trained generator, mapping high-dimensional sentiment proxies to a single low-dimensional sentiment factor. Unlike direct deep learning regression models, which often overfit short-term fluctuations, the adversarial training mechanism in GANs learns robust latent information while filtering noise.

② To further enhance stability, multiple GAN models are trained, and their outputs are averaged. A comparative study on ensemble sizes (3, 5, 10) indicates that using 5 models achieves the optimal trade-off between accuracy and computational efficiency. Variance analysis confirms the robustness of this approach.

The pseudocode of the GANs is listed in Algorithm 4 below.

Algorithm 4. Pseudocode of GANs.

Require: financial feature matrix (X), learning rate (α), batch size (B), hidden layer size (H), max epochs (T)
function initializeGAN (H)
    $G \leftarrow$ Three-layer neural network with hidden size (H)
    $D \leftarrow$ Fully connected neural network
    return G, D
end function
function trainGAN $(G, D, X, \alpha, \beta, T)$
    for epoch = 1 to T **do**
        Sample minibatch (X_B) from (X)
        $S_G \leftarrow G$. Extttforward (X_B)
        $L_D \leftarrow$ extttDiscriminatorLoss (D, S_G)
        Update D using Adam optimizer with (α)
        $L_G \leftarrow$ extttGeneratorLoss (G, D, X_B)
        Update G using Adam optimizer with (α)
    end for
    return Trained G as (S^{GAN})
end function

3.4.4. Innovation of GANs Applied to Sentiment Indicator Construction

Compared with traditional linear dimensionality reduction techniques, GANs show unique advantages in sentiment indicator construction. Firstly, GANs are able to automatically learn and capture complex nonlinear dependencies between sentiment agents without being restricted by the assumption of linearity. Moreover, GANs have dynamic weight adjustment. For example, through adversarial training, GANs are able to dynamically adjust feature weights according to the contribution of each sentiment agent to the excess return prediction to dynamically adjust the feature weights. Last but not least, GANs are capable of noise filtering. The presence of the discriminator ensures that the generated sentiment factors have high signal-to-noise ratios, effectively filtering the market noise. These innovative features make GANs particularly suitable for capturing the complex dynamics of investor sentiment.

In this study, we not only simply apply GANs to financial data, but also design algorithmic innovations to address the specific needs of investor sentiment extraction. We developed a specialized GAN architecture. It includes an innovative dual-loss function design that combines adversarial loss and regression constraints to ensure that the extracted sentiment factors both conform to statistical laws and maintain economic explanatory power. Additionally, our GAN architecture contains label smoothing and adaptive dropout strategies for financial time series, which effectively prevents pattern crashes and overfitting. Moreover, a feature importance assessment based on a gradient analysis mechanism reveals the dynamic changes in market sentiment structure. These innovative designs enable our GAN model to extract comprehensive sentiment indicators with strong predictive power from high-dimensional sentiment proxies.

4. Empirical Results

4.1. Single-Factor Regression Analysis

4.1.1. Adversarial Training Performance of GANs

We first verify the performance of the GANs. In Figure 7, the consistent decline in the discriminator's loss suggests a progressive enhancement in its capacity to distinguish real

data from generated data. The generator's loss decreases rapidly at first, but then the curve flattens out and eventually stabilizes. This indicates that the generator loss stabilizes as training progresses and that it is able to generate relatively realistic data. Figure 8 shows that as training proceeds, the generator succeeds in gradually converging the distribution of generating factors from a random distribution to a more regular distribution. This trend is one of the important signs of successful GAN training.

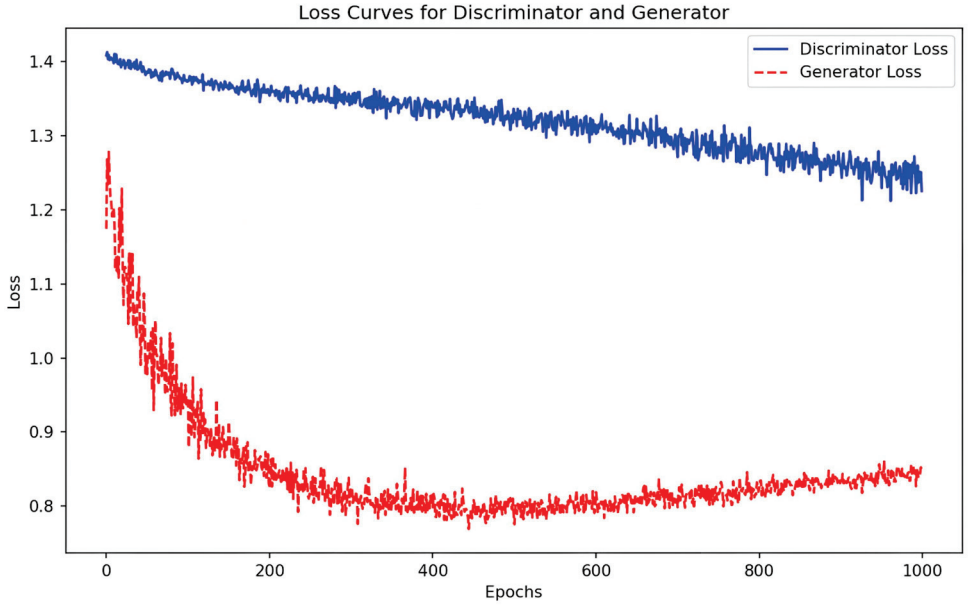


Figure 7. Loss curves for discriminator and generator.

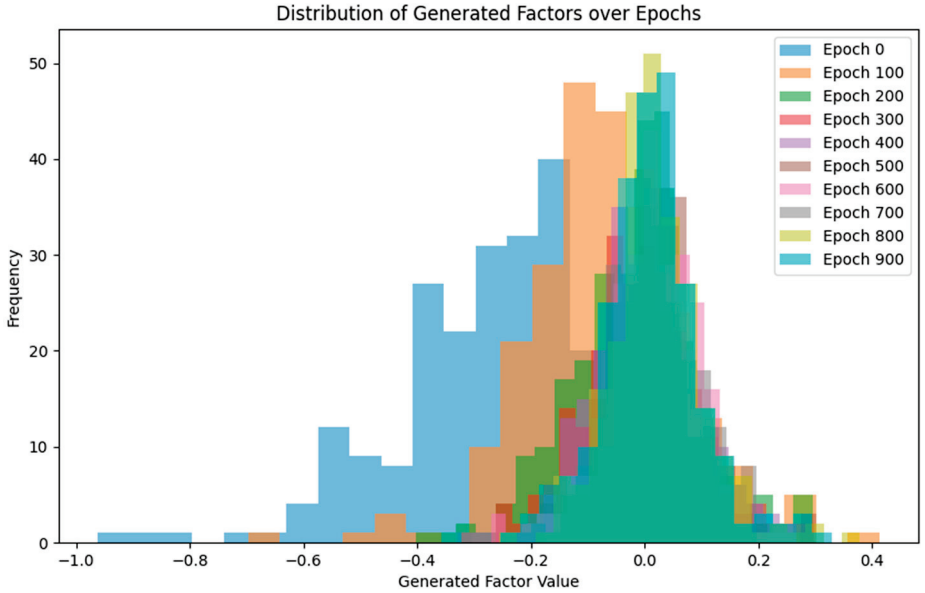


Figure 8. Distribution of generated factors over epochs.

4.1.2. Single-Factor Regression of Four Investor Sentiments

We measure the predictive power of S^{PCA}, S^{PLS}, S^{GA}, and S^{GAN} for Chinese A-share market returns, respectively. Given the extensive and well-established literature on predictive regression with a single factor, we begin our analysis by considering single-predictor regression in the context of prior investigations.

The model constructed to evaluate the predictability of investor sentiment indicators in this subsection is as follows:

$$R_{t+1}^{m} = \alpha + \beta S_t^X + \varepsilon_{t+1} X = PCA, PKS, GA, GAN \tag{7}$$

R_{t+1}^{m} represents the excess market returns at $t + 1$ month and is the lagged sentiment indicator. The predictive ability of S_t^X is assessed through the estimated coefficient of the slope β, along with the corresponding t-statistics.

As demonstrated in Table 5, S^{GAN} shows strong in-sample forecasting performance for the Chinese A-share market, with an estimated efficient 0.897, and achieves a significance level of 1%. Moreover, the GAN model achieves a respectable goodness of fit, as evidenced by an R^2 of 15.3%, indicating a solid explanatory power for stock market behavior.

Table 5. In-sample prediction effect of four different investor sentiments.

Variables	Coefficient	t-Statistics	R^2 (%)
S^{GAN}	0.897 ***	13.982	15.3
S^{GA}	1.000 ***	6.010	12.8
S^{PLS}	−0.031 ***	−5.242	10.7
S^{PCA}	0.032	0.496	−8.7

(Notes: Table 5 gives the regression results of S^{GAN}, S^{PLS}, S^{GA} (developed by GA method), and S^{PCA}, respectively. The specific indicators include coefficient, t-statistics, and R^2. Statistical significance levels are denoted by asterisks. *** at the 1% significance level. The period is from February 2003 to September 2023).

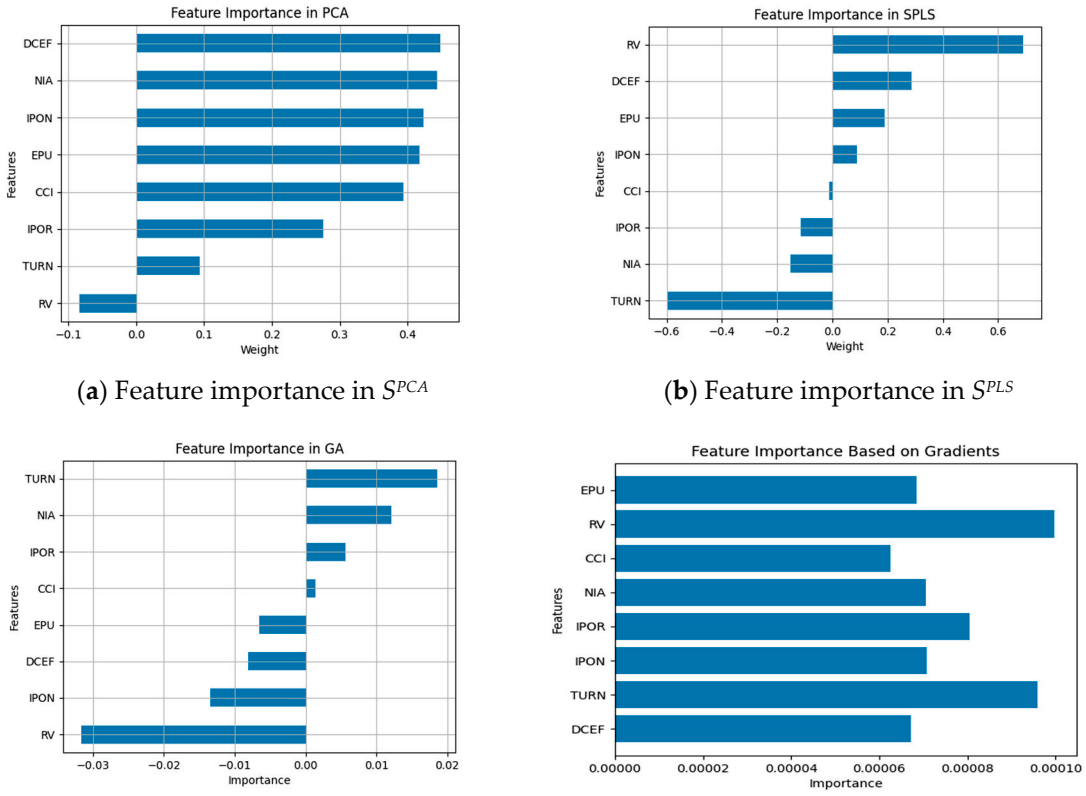
Similarly, S^{PLS} displays a negative but significant forecasting power, with an estimated slope of −0.031. This indicator also offers a reasonably good fit, with an R^2 value of 10.0%. Also, the coefficient of S^{GA} is 1.000 with a significance level of 1%, and the R^2 is 12.8%. In contrast, according to row 5 of Table 5, we observe that the in-sample predictive ability of S^{PCA} is considerably weaker, with an estimated parameter that is statistically not statistically significant, suggesting it lacks explanatory power regarding excess market returns. Nonetheless, PCA is still a valid benchmarking method, and its weaker performance compared to newer methods such as GANs highlights the strength of the GAN method in capturing sentiment signals.

4.1.3. Analysis of Feature Importance

To quantify feature importance with precision, we employ gradient attribution analysis in our GAN model. This technique measures the gradient of the output with respect to each input feature, providing a numerical assessment of each feature's contribution to the model's decisions. The gradient values are normalized to a scale of 0–1, with higher values indicating stronger influence on model outcomes. This quantitative approach complements our qualitative analysis and provides a rigorous foundation for identifying the most significant features in our framework.

As shown in Figure 9, traditional dimensionality reduction methods (e.g., PCA) make most proxies contribute almost the same to the sentiment measure and thus may contain noise or irrelevant information. In contrast, S^{PLS}, S^{GA}, and S^{GAN} could optimize sentiment extraction by selectively weighting features according to their contribution to excess return predictability. However, S^{PLS} and S^{GA} are substantially dependent on a few agents. However, we use the corresponding gradient size of each agent in the GANs to measure its contribution to S^{GAN}. Sentiment proxies reflecting liquidity conditions (e.g., TURN, RV) and speculative activities (e.g., IPOR) have high correlations with the extracted factors, suggesting that they play a dominant role in influencing market sentiment. Meanwhile,

indicators of macroeconomic uncertainty (e.g., EPU, CCI) have less impact on the financial market, but still act together with other indicators on the stock market.



(**a**) Feature importance in S^{PCA}

(**b**) Feature importance in S^{PLS}

(**c**) Feature importance in S^{GA}

(**d**) Feature importance in S^{GAN}

Figure 9. Feature importance in S^{PCA}, S^{PLS}, S^{GA}, and S^{GAN}.

Unlike traditional methods such as PCA that assign weights statically, S^{GAN} is able to dynamically adjust the importance of each sentiment proxy in response to changes in market conditions. The GANs assign higher weights to liquidity-related indicators and speculative activity indicators, while maintaining the moderate influence of macroeconomic uncertainty indicators, creating a more balanced distribution of feature importance. This adaptive feature weight distribution mechanism is a key factor in the superior forecasting performance of S^{GAN} and represents an important innovation in sentiment indicator construction techniques.

In order to systematically assess the sensitivity of S^{GAN} to each emotional agent variable, we perform a feature leave-one-out test. This test excludes eight investor sentiment agents one by one, trains multiple S^{GAN} variants, and compares the changes in their predictive performance. The specific testing results are shown in Table 6.

Based on the results of the feature leave-one-out test, we find that S^{GAN} varies in its dependence on each of the sentiment proxy variables, but shows strong feature robustness. The exclusion of RV had the greatest impact on the model, resulting in a decrease in R^2 of 0.0506, followed by IPOR and TURN, which decreased R^2 by 0.0458 and 0.0424, respectively. In contrast, the exclusion of DCEF, IPON, and EPU had less impact on the model performance. These results indicate that S^{GAN} successfully incorporates multidimensional sentiment information, with particular emphasis on market volatility and speculative activity-related indicators, while not overly relying on any single variable, and maintains a stable predictive ability even in the absence of certain sentiment proxies, which validates the robustness and adaptability of the model.

Table 6. Leave-one-out tests of $S^{GAN.}$

	R^2 (%)	Changes in R^2 (%)
Exclude DCEF	0.1486	0.0049
Exclude TURN	0.1111	0.0424
Exclude IPON	0.1495	0.0040
Exclude IPOR	0.1077	0.0458
Exclude NIA	0.1132	0.0403
Exclude CCI	0.1384	0.0151
Exclude RV	0.1029	0.0506
Exclude CPU	0.1474	0.0061

(Notes: Table 6 gives the leave-one-out tests of S^{GAN} developed by GAN method. The specific indicators include R^2 (%) and changes in R^2 (%). The period is from February 2003 to September 2023.)

4.1.4. Backtesting of Predictability

We also plot backtests of the market excess returns predicted by four different investor sentiment forecasts. In Figure 10, plot (a) shows the backtest of predicted returns using PCA, and it can be seen that S^{PCA}s are not effective in making predictions about the actual market conditions. The S^{PLS} and S^{GA} backtest results in plot (b) and plot (c) show higher instability. In plot (d), the blue line represents the backtest results of the actual market, which is more volatile, especially in the early and middle periods, when the net value experiences rapid growth and fallback, with higher volatility. The red line represents the GAN-based investor sentiment strategy, which is more volatile in the early stages and shows some profit potential and a relatively stable growth trend. Furthermore, the GAN-based investment strategy shows relatively positive returns over the backtesting period, with an annualized return of 8.80%, implying a return of approximately 8.80% per annum. Moreover, using the 1-year China Treasury yield as the risk-free rate of return, the strategy's Sharpe ratio was 1.35, which is a reasonable return per unit of risk and suggests that the strategy's risk-adjusted level of return is somewhat effective.

4.2. Out-of-Sample Tests

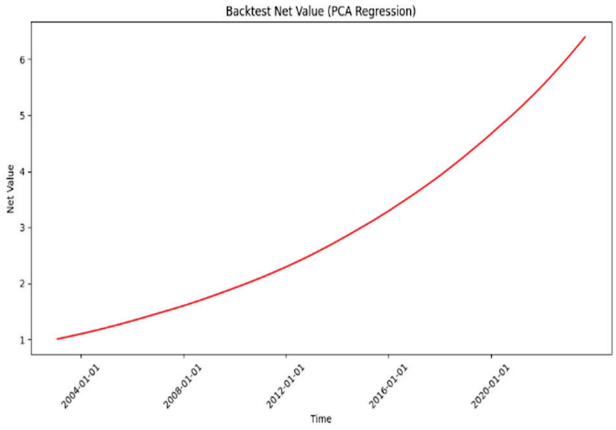
4.2.1. Out-of-Sample Tests for PCA, PLS, GA, and GANs

This section focuses on the performances of the S^{PLS}, S^{GA}, and S^{GAN} in out-of-sample forecasting. Huang et al. [32], and others, demonstrate that out-of-sample testing effectively mitigates biases, providing a more robust evaluation of model performance.

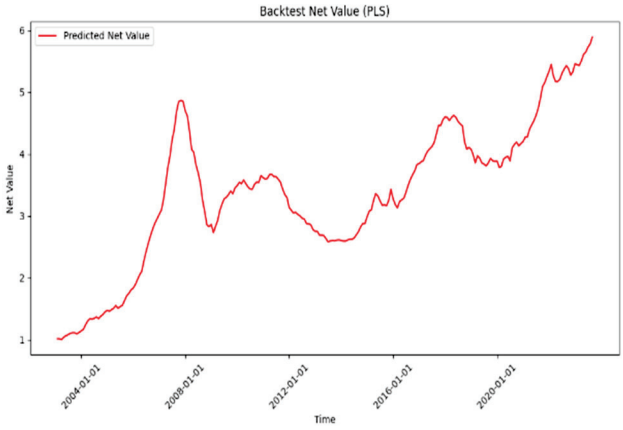
The full observation window is divided into two intervals. The calibration period extends from February 2003 to August 2019, followed by a four-year validation window (2019–2023) for performance verification. The parameter p is designated as the fixed length of the initial training dataset. This model is then applied to predict excess market returns at subsequent time intervals $t = p + 1, p + 2$. For the time $p + 1$, investor sentiment indices are derived from the initial training dataset to forecast these excess returns.

$$E(R^m_{t+1}) = \hat{\alpha}_t + \hat{\beta}_t S^X_t, t = p, p+1, \cdots, T-1, X = PCA, PLS, GA, GAN \tag{8}$$

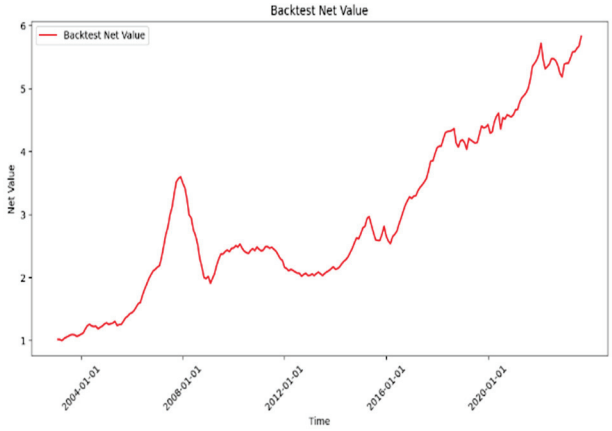
where $\hat{\alpha}_t$ and $\hat{\beta}_t$ are estimated by using regression $\{R_{s+1}\}_{s=t-p+1}^{t-1}$ on a constant and investor sentiments $\{S^X_s\}_{s=t-p+1}^{t-1}$. Additionally, S^X_t and S^X_s are derived exclusively from data spanning periods $t - p + 1$ through $t - 1$. Consequently, our estimation of excess market returns at period t relies solely on information available through time t.



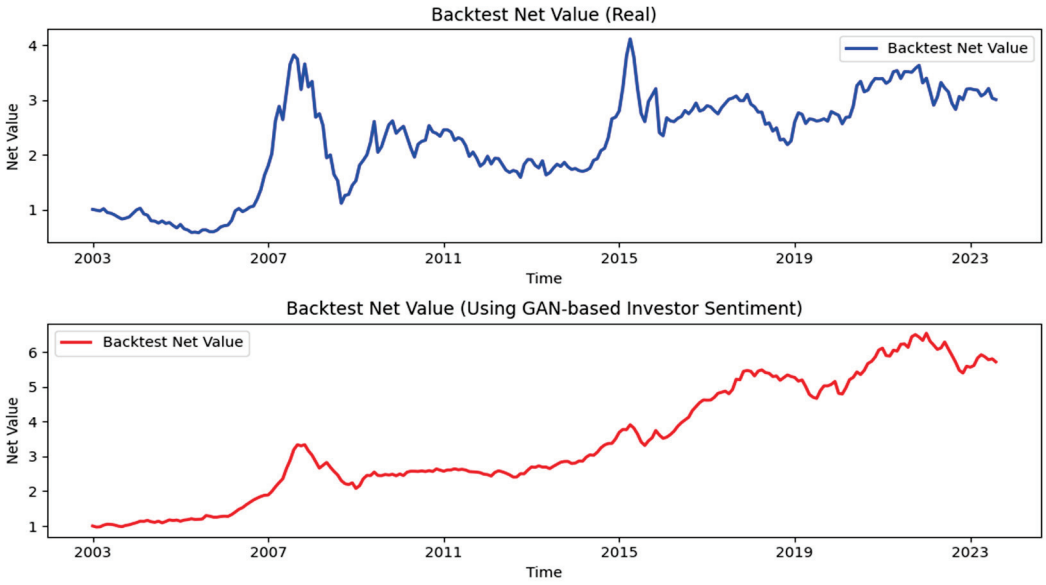
(**a**) Backtest net value of S^{PCA}



(**b**) Backtest net value of S^{PLS}



(**c**) Backtest net value of S^{GA}



(**d**) Backtest net value of real return and predicted return of S^{GAN}

Figure 10. The backtest net value of real returns for S^{GAN}, S^{PCA}, S^{PLS}, and S^{GA}.

To assess the forecasting performance, our paper utilizes the R^2_{OS} statistic and the MSFE-adjusted statistic, both of which are commonly employed in the literature. The R^2_{OS} is defined as follows:

$$R^2_{OS} = 1 - \frac{\sum_{t-p}^{T-1} (R^m_{t+1} - E(R^m_{t+1}))^2}{\sum_{t-p}^{T-1} (R^m_{t+1} - \overline{R}^m_{t+1})^2} \tag{9}$$

$$\overline{R}^m_{t+1} = \frac{1}{t} \sum_{s=1}^{t} R_s \tag{10}$$

Here, R^m_{t+1} represents the actual excess market return at the time $t+1$, while $E(R^m_{t+1})$ and $\overline{R}^m_{t+1}$ denote the estimated excess market returns and the historical average benchmark, respectively. As outlined by Neely and others [33], the R^2_{OS} statistic quantifies the relative decrease in MSFE compared with the benchmark. To statistically validate forecasting efficacy, the MSFE-adjusted measure is further applied, assessing sentiment-driven models against the reference framework.

Hou emphasizes that historical averages provide an exceptionally rigorous benchmark in out-of-sample analyses, with most economic predictors failing to surpass this standard [34]. We conclude that our prediction $E(R^m_{t+1})$ is superior to the historical means if $R^2_{OS} > 0$. Anticipated reductions in MSFE for our proposed framework relative to established benchmarks would demonstrate predictive superiority over historical mean returns in comparative forecasting accuracy assessments.

Table 7 reports the out-of-sample predictability of different investor sentiments for the excess market returns. Table 7 indicates that S^{PLS} and S^{GA} have significantly great R^2_{OS} values, which are 10.60% and 16.37%, respectively, and the corresponding MSFE-adjusted statistics are 0.008 and 0.007, respectively, and are statistically significant at a 1% level. S^{GAN} brings an R^2_{OS} value of 25.40%, and the MSFE-adjusted measure yields a value of 0.009, attaining significance at the 1% level. The R^2_{OS} values of S^{PCA}, MPCI, MSPI, and MSLI are all less than zero, and their MSFE-adjusted statistics are not statistically significant. According to Huang et al. [32], the R^2_{OS} value of 0.5% could create considerable economic value. Therefore, S^{GAN} demonstrates strong performance in forecasting excess market returns and might be effectively applied in real-world financial practice. Furthermore, by comparing the R^2_{OS} values among S^{PLS}, S^{GA}, and S^{GAN}, it suggests that S^{GAN} has better predictability for excess returns.

Table 7. Out-of-sample predictability.

Predictor	R^2_{OS} (%)	MSFE-Adj
S^{GAN}	25.40	0.009 ***
S^{GA}	16.37	0.007 ***
S^{PLS}	10.60	0.008 ***
S^{PCA}	−8.67	0.966
MPCI	−9.00	0.969
MSPI	−7.20	0.953
MSLI	−7.27	0.954

(Notes: This table shows the out-of-sample tests for S^{PLS}, S^{GA}, S^{GAN}, S^{PCA}, MPCI, MSPI, and MSLI. The relevant statistics include the R^2_{OS} and MSFE-adjusted statistics. The period of training is from February 2003 to August 2019. The period of validation is from September 2019 to September 2023. Statistical significance levels are denoted by asterisks. *** at the 1% significance level.)

4.2.2. Rolling Forecasts of S^{GAN}

When reaching the final time period of testing, Figure 11 represents that the loss variance of the test set stays at a low level, indicating that the performance on the test

set is stable and may have reached a reasonable level of generalization. We also use rolling forecasts to test the model for overfitting. Figure 12 represents the rolling window prediction visualization results. Under the rolling window, the average MSE value of the training set is 0.00939, while the average MSE value of the test set is 0.00305. From these data, we find that the difference in MSE between the training set and the test set is small, and both of them are relatively low, which suggests that the model performs well in terms of generalization ability, and there is no sign of overfitting. The experimental results show that when the dropout mechanism is added to the GANs, it can effectively avoid pattern collapse and overfitting problems, ensuring its robust performance on out-of-sample data.

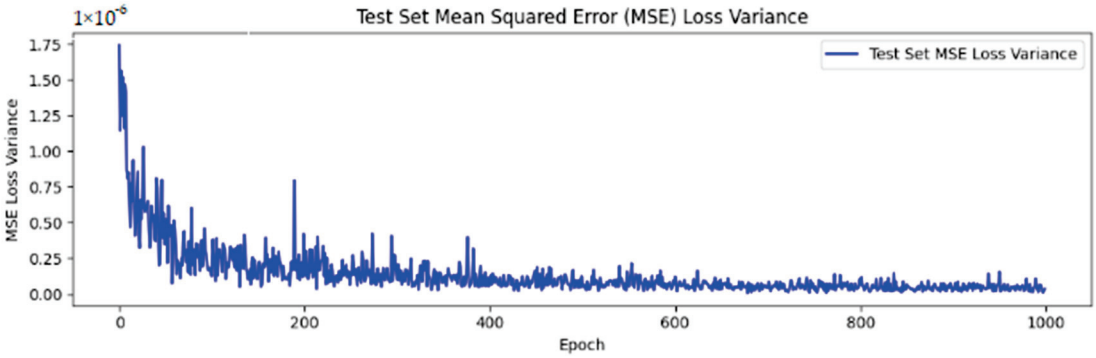


Figure 11. Test set mean squared error loss variance.

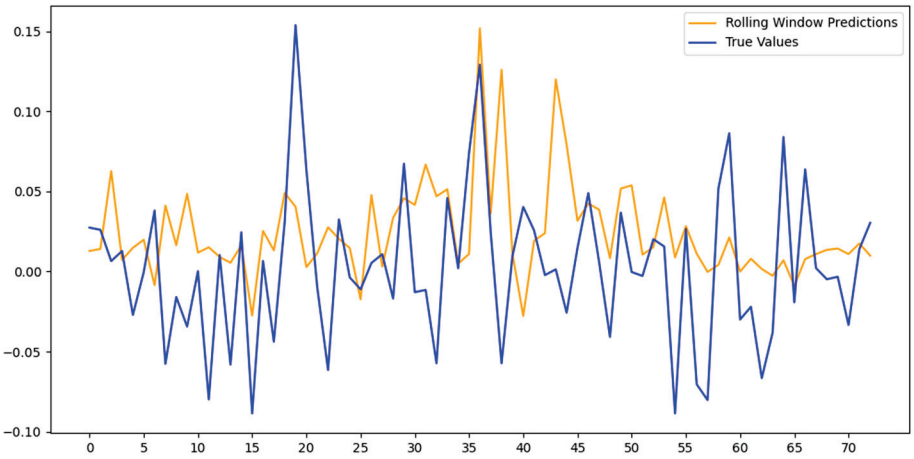


Figure 12. The rolling window prediction of test set.

4.3. Predictability for Different Industries

The previous sections demonstrated the notable impacts of investor sentiment on overall market dynamics. Theoretically, securities characterized by higher valuation complexity and arbitrage constraints are likely more susceptible to sentiment-driven fluctuations relative to other equities [35]. Song et al. find that PCA-based investor sentiment can significantly impact the excess return of different industries [36]. Following the method of Song and Yu, we continue to test the influence of S^{GAN} on the returns in different industries. In this section, we examine the predictive capacities of investor sentiments through systematic selection of seven sectoral groupings within China's A-share market. These seven industries are selected from the primary industries classified by the CSRC, including the agriculture index, catering index, financial index, construction index, and other primary sector indices.

In Table 8, the return predictability of S^{GAN} varies across industries, with the significance at the 1% level, which confirms that S^{GAN} has the high predictive ability of the

corresponding returns across industries. More importantly, we find that the return prediction ability of S^{GAN} is more prominent for the financial sector, where the corresponding R^2 is the largest among the seven sectors. The financial sector's high sensitivity to changes in market sentiment stems mainly from its close relationship with market speculation, leveraged trading, and policy sensitivity. The sentiment S^{GAN} prioritizes variables that reflect market speculation and risk appetite, such as TURN and RV. These proxies are directly related to investor trading activity and are particularly significantly impacted. This inference is consistent with the results presented in previous studies by Song and Yu [37]. However, in general, S^{GAN} outperforms the other three sentiments in predicting excess market returns, and we infer that employing S^{GAN} could generate enhanced economic utility for financial sector applications and advanced theoretical frameworks.

Table 8. Predictability for different industries.

	S^{PCA}			S^{GA}		
Industry	**Coefficient**	**t-Stat**	**R^2 (%)**	**Coefficient**	**t-Stat**	**R^2 (%)**
Agriculture	−0.003	−0.034	−0.8	1.010	2.991 **	6.7
Construction	−0.002	−0.089	−0.8	1.001	3.764 ***	10.2
Retail	−0.004	−0.055	−1.2	1.000	4.092 ***	11.8
Catering	−0.002	−0.046	−0.8	1.003	2.960 **	8.0
Finance	−0.007	−0.101	−0.9	1.014	5.074 ***	17.1
Real Estate	−0.003	−0.026	−0.8	1.010	5.281 ***	16.5
Mining	−0.003	−0.038	−0.9	1.003	3.463 **	8.8
	S^{PLS}			S^{GAN}		
Industry	**Coefficient**	**t-Stat**	**R^2 (%)**	**Coefficient**	**t-Stat**	**R^2 (%)**
Agriculture	0.030	2.325	4.1	1.672	14.127 ***	11.2
Construction	0.052	3.130 **	7.3	1.394	11.179 ***	14.6
Retail	0.035	3.183 **	7.5	1.261	16.346 ***	17.7
Catering	−0.026	−2.399	5.2	1.470	12.149 ***	9.6
Finance	0.045	4.078 ***	11.7	1.360	13.138 ***	21.9
Real Estate	0.031	3.497 **	8.0	1.401	11.855 ***	17.0
Mining	0.037	2.745 *	5.7	1.506	10.839 ***	10.2

Notes: This table represents the predicting power of S^{PCA}, S^{PLS}, S^{GA}, and S^{GAN} for the seven in-sample industry indices, respectively. The coefficient, t-statistic, and R^2 are reported. Statistical significance levels are denoted by asterisks: * indicates significance at the 10% threshold, ** at 5%, and *** at the 1% significance level. The period is from February 2003 to September 2023.

5. Robustness Check

5.1. Multi-Factor Predictive Regression

In the previous section, we observed that the S^{PLS}, S^{GA}, and S^{GAN} showed significant return predictability, with S^{GAN} exhibiting particularly promising results. However, it is possible that certain economic factors within these indices could also predict returns, potentially affecting the robustness of our findings. To address this, our study investigates the macroeconomic variables' predictive capacity and examines whether investor sentiments retain significant explanatory power for excess market returns after controlling for fundamental economic determinants. In this paper, four economic factors are incorporated into the analysis to examine their predictive capacity and evaluate the four sentiments' predicting abilities under the control of these factors. The four economic factors include the macroeconomic prosperity consensus index (MPCI), macroeconomic sentiment proximity index (MSPI), macroeconomic sentiment lag index (MSLI), and consumer sentiment index (CSI).

Firstly, we utilize the single-factor regression analysis:

$$R_{t+1}^{m} = \alpha + \beta E_t^k + \varepsilon_{t+1}, k = 1, 2, 3, 4 \tag{11}$$

where E_t^k represents the kth economic factor raised in Section 2.

Panel A of Table 8 provides the estimated results for the four economic factors, in which we find that the MPCI, MSPI, MSLI, and CSI report estimation results with R^2 values of 0.9%, 5.1%, 3.0%, and 0.3%, respectively.

Secondly, we assess whether the predictive capacity of S^{PLS}, S^{GA}, and S^{GAN} remains robust after controlling for macroeconomic variables. For this purpose, predictive regression models are constructed, incorporating both investor sentiment indices and economic factors E_t^k:

$$R_{t+1}^m = \alpha + \beta E_t^k + \gamma S_t^X + \varepsilon_{t+1}, k = 1, 2, 3, 4, 5, X = PCA, PLS, GA, GAN \quad (12)$$

Panels B and C of Table 8 report the results of S^{PLS} and S^{GA}, respectively. The corresponding coefficients for S^{GA} and S^{PLS} are significant, implying that S^{GA} and S^{PLS} could still significantly predict excess market returns after adding economic factors. Panel E of Table 8 shows the poor predictability of S^{PCA} after adding the four economic factors.

Panel D of Table 9 presents the regression results for S^{GAN}, showing that the corresponding slope remains substantial and statistically significant even after accounting for economic factors. By comparing the results in other panels, we find that the R^2 of S^{GAN} in the tests on forecasting excess market returns is higher than the R^2 of S^{PLS} and S^{GA} in similar tests here. These results suggest that S^{GAN} contains more predictive information about excess market returns than the others.

Table 9. Predicting excess returns with macroeconomic variables.

Panel A: single-factor results $R_{t+1}^m = \alpha + \beta E_t^k + \varepsilon_{t+1}$					
Variables	β (%)	t-statistics	R^2 (%)		
MPCI	0.060 ***	1.512	0.9		
MSPI	0.240 ***	3.630	5.1		
MSLI	−0.150 *	−2.775	3.0		
CSI	0.010	0.814	0.3		
Panel B: S^{PLS} and economic factors results $R_{t+1}^m = \alpha + \beta E_t^k + \gamma S_t^{PLS} + \varepsilon_{t+1}$					
Variables	γ (%)	t-statistics	β (%)	t-statistics	R^2 (%)
MPCI	−3.27 ***	−5.468	−0.200	−1.792	11.2
MSPI	−3.18 ***	−5.119	−0.060	−0.335	10.1
MSLI	−2.96 ***	−4.816	−0.150	−0.995	10.4
CSI	−3.30 ***	−5.481	0.080	1.684	11.1
Panel C: S^{GA} and economic factors results $R_{t+1}^m = \alpha + \beta E_t^k + \gamma S_t^{GA} + \varepsilon_{t+1}$					
Variables	γ (%)	t-statistics	β (%)	t-statistics	R^2 (%)
MPCI	102.59 ***	6.157	−0.180	−1.617	13.7
MSPI	100.80 ***	5.890	−0.040	−0.207	12.8
MSLI	96.15 ***	5.699	−0.190	−1.308	13.4
CSI	99.66 ***	5.971	0.020	0.391	12.9
Panel D: S^{GAN} and economic factors results $R_{t+1}^m = \alpha + \beta E_t^k + \gamma S_t^{GAN} + \varepsilon_{t+1}$					
Variables	γ (%)	t-statistics	β (%)	t-statistics	R^2 (%)
MPCI	131.38 ***	14.059	−0.140	−1.606	15.4
MSPI	132.24 ***	13.919	−0.110	−0.790	15.0
MSLI	129.76 ***	13.705	−0.110	−0.960	15.0
CSI	130.92 ***	13.937	0.002	0.414	14.9

Table 9. *Cont.*

Panel E: S^{PCA} and economic factors results $R^m_{t+1} = \alpha + \beta E^k_t + \gamma S^{PCA}_t + \varepsilon_{t+1}$					
Variables	γ (%)	t-statistics	β (%)	t-statistics	R^2 (%)
MPCI	0.031	0.465	0.059 ***	1.512	0.9
MSPI	0.031	0.467	0.240 ***	3.630	5.1
MSLI	0.029	0.492	−0.150 *	−2.775	3.0
CSI	0.030	0.501	0.010	1.814	0.3

(Notes: This table evaluates in-sample predictive performance across methodological configurations for forecasting excess returns, employing four macroeconomic indicators (MPCI, MSPI, MSLI, CSI) as baseline predictors. Panel A establishes benchmark results using individual economic variables, while Panels B-E progressively integrate these with alternative sentiment quantification techniques: PLS-derived indices (Panel B), genetic algorithm-optimized metrics (Panel C), GAN-generated composites (Panel D), and PCA-based proxies (Panel E). Statistical significance levels are denoted by asterisks: * indicates significance at the 10% threshold, and *** at the 1% significance level. The period is from February 2003 to September 2023).

5.2. Economic Significance Analysis

5.2.1. Predictability over Business Cycle

The predictability of investor sentiment has been shown to vary across different time periods [37]. Consequently, we perform the robustness check to evaluate whether the principal findings maintain statistical and economic significance in varying periods. The global financial market experienced a significant shock in 2008, known as the subprime mortgage crisis. The full sample is divided at the onset in December 2007. Similarly, the 2015 to 2016 Chinese stock market turbulence had a profound impact on investor confidence in future market conditions.

We run the regression model in Section 4.1 for S^{PCA}, S^{PLS}, S^{GA}, and S^{GAN} again over the four time periods. The temporal analysis employs systematically segmented intervals to evaluate sentiment predictability under distinct market regimes. For the 2008 financial crisis, the sample is stratified into pre-crisis (February 2003–December 2007) and post-crisis (January 2008–September 2023) phases. Similarly, China's 2015 equity market correction is analyzed through pre-turbulence (February 2003–June 2015) and post-turbulence (July 2015–September 2023) windows. Table 9 presents the comparative predictive efficacy of sentiment metrics across these temporally stratified intervals, revealing structural variations in forecasting capacity during crisis versus noncrisis market states.

Table 10 provides a comparative analysis of different predictors. For example, before the subprime crisis, S^{PLS} exhibited significant positive predictive power, but after the crisis, its predictive ability turned negative, and its explanatory power decreased substantially, indicating that the market's reliance on this variable weakened in the face of the crisis. S^{GAN} reflects the dramatic shift in market mood, showing a sharp drop in sentiment that corresponded with the market's steep decline. The recovery seen later is captured by the rebound in sentiment, demonstrating S^{GAN}'s ability to predict the market's reaction to global financial events and policy interventions.

A similar pattern is observed in the period surrounding the Chinese stock market turbulence. S^{GAN} exhibited greater robustness, as its predictive ability remained stable and even improved after the turbulence, with a significant increase in explanatory power. These results indicate that S^{GAN} demonstrates a high degree of robustness during both crises, effectively responding to external shocks, whereas S^{PLS} and S^{GA} show greater variability in their predictive performances across different stages.

Table 10. Predictability before and after the financial crisis.

Predictor	Coefficient	t-Statistics	R^2 (%)
Before the sub-prime crisis			
S^{PCA}	0.034	0.511	−7.9
S^{PLS}	0.020 **	3.274	16.1
S^{GA}	1.000 ***	4.077	22.6
S^{GAN}	0.316 ***	8.090	28.4
After the sub-prime crisis			
S^{PCA}	0.032	0.498	−7.8
S^{PLS}	−0.019 ***	−4.271	8.9
S^{GA}	1.000 ***	6.161	16.9
S^{GAN}	0.239 ***	14.932	19.7
Before China's 2015 equity market correction			
S^{PCA}	0.029	0.612	−8.0
S^{PLS}	0.028 ***	4.823	13.7
S^{GA}	1.000 ***	5.949	19.4
S^{GAN}	0.352 ***	9.366	22.5
After China's 2015 equity market correction			
S^{PCA}	0.030	0.488	−8.1
S^{PLS}	−0.018 ***	−5.191	21.9
S^{GA}	1.000 ***	6.195	28.4
S^{GAN}	0.196 ***	10.879	30.3

(Notes: This table reports robustness check results and predictivity results for the regression model in Section 4.1. For the 2008 financial crisis, the sample is stratified into pre-crisis (February 2003–December 2007) and post-crisis (January 2008–September 2023) phases. Similarly, China's 2015 equity market correction is analyzed through pre-turbulence (February 2003–June 2015) and post-turbulence (July 2015–September 2023) windows. The specific indicators include coefficient, t-statistics, and R^2. Statistical significance levels are denoted by asterisks: ** at 5%, and *** at the 1% significance level).

5.2.2. Predictability During Bull Market, Bear Markets, and Turning Points

As a measure of market sentiment, the indicator exhibits strong indicative power in bull–bear market transitions. Specifically:

(1) Bull Market: At the early stage of a bull market, S^{GAN} exhibits a gradual upward trend, indicating a steady improvement in investor sentiment. As the market reaches its late phase, S^{GAN} experiences a sharp surge, suggesting potential overheating and speculative bubbles.

(2) Bear Market: In the initial phase of a bear market, S^{GAN} declines rapidly, reflecting a significant deterioration in investor confidence. Towards the end of the bear cycle, S^{GAN} remains at a low level but gradually recovers as signs of market stabilization emerge. A notable example is the 2008 global financial crisis, where S^{GAN} plummeted to a trough before gradually rebounding with the onset of economic recovery in 2009.

(3) Turning Points: S^{GAN} tends to lead market return fluctuations at critical turning points, meaning that when it reaches an extreme high or low, a market trend reversal often follows. This characteristic makes S^{GAN} a valuable reference for investment strategies. For instance, a downturn from a high S^{GAN} level may signal rising market risks, prompting a more cautious approach, whereas an upward shift from a low S^{GAN} level may indicate a favorable entry point for investors.

5.2.3. Predictability During Bear Markets

In the Chinese stock market, bear markets are usually triggered by a range of economic, political, and market factors [36]. In the past few years, the Chinese stock market has

experienced several notable bear markets, particularly in 2015, 2018, and 2021 to 2023 [38]. Market behavior during these bear markets provides an important opportunity to test the validity of investor sentiment indicators [39].

As can be seen in Table 11, S^{GAN} has demonstrated strong forecasting ability during different bear market periods. During the crash between June 2015 and February 2016, the predictive performance of S^{GAN} is impressive and significant, with an R^2 of 68.4%, demonstrating the model's high ability to explain excess market returns. In contrast, S^{PCA} has no predictive power in almost all the time periods. S^{PLS} performs relatively weakly, especially during the same period, with an R^2 of 61.3% and a negative coefficient, suggesting that its forecasts are not as good during this period. The performance of S^{GA} is relatively robust, with the significant coefficient and high R^2, but still not as good as that of S^{GAN}. In addition, during the other periods of the bear market, in 2018 and from 2021 to 2023, S^{GAN} still maintains a high R^2 value of 71.8% and 47.2%, respectively, showing its robustness and consistency across market cycles. Overall, S^{GAN} performs well in all bear market periods analyzed, proving its strong predictive ability as a sentiment indicator in extreme market conditions.

Table 11. Predictability during bear markets.

Predictor	Coefficient	t-Statistics	R^2 (%)
June 2015 to February 2016			
S^{PCA}	0.028	0.488	0.8
S^{PLS}	−0.071 *	−3.696	61.3
S^{GA}	1.000 ***	13.339	65.2
S^{GAN}	0.589 ***	19.353	68.4
January 2018 to December 2018			
S^{PCA}	0.031	0.546	0.8
S^{PLS}	−0.026 ***	−4.678	67.6
S^{GA}	1.000 ***	12.479	70.4
S^{GAN}	0.320 ***	9.496	71.8
January 2021 to September 2023			
S^{PCA}	0.030	0.522	0.8
S^{PLS}	−0.025 ***	−4.325	36.4
S^{GA}	1.000 ***	5.083	43.7
S^{GAN}	0.256 ***	6.625	47.2

(Notes: This table reports robustness check results and predictivity results for the regression model in Section 4.1. The time periods include June 2015 to February 2016, January 2018 to December 2018, and January 2021 to September 2023. The specific indicators include coefficient, t-statistics, and R^2. Statistical significance levels are denoted by asterisks: * indicates significance at the 10% threshold, and *** at the 1% significance level).

5.2.4. Which Market Behaviors (e.g., Speculation, Market Bubbles) Influence S^{GAN} Dynamics?

As an investor sentiment index, S^{GAN} captures various behavioral patterns in financial markets. It demonstrates notable sensitivity to specific market conditions, particularly in the following scenarios:

(1) Speculation: Empirical evidence suggests that one of S^{GAN}'s key sentiment factors is market trading activity (TURN), which tends to increase during periods of heightened speculative behavior. For instance, in bullish markets, retail investors typically engage in higher trading volumes, leading to a surge in overall market sentiment, which is subsequently reflected in S^{GAN}'s upward movement.

(2) Market Bubbles: S^{GAN} is also effective in identifying market bubble formations. During major bull runs, such as those in 2007 and 2015 of the Chinese A-share market,

S^{GAN} recorded significant increases, signaling extreme investor optimism. Notably, before a bubble bursts, S^{GAN} often exhibits an early downward trend.

(3) Flight to safety: During periods of heightened market uncertainty, such as the 2008 global financial crisis and the 2020 COVID-19 pandemic, S^{GAN} experienced a pronounced decline, reflecting deteriorating investor sentiment and a shift towards safer assets. This trend aligns with the movement of the VIX index (volatility index), reinforcing S^{GAN}'s ability to capture risk aversion behavior under financial distress.

5.2.5. How Do S^{GAN}'s Core Sentiment Factors Relate to Investor Behavior?

The key sentiment factors extracted by S^{GAN} using deep learning techniques exhibit strong associations with investor behavior in financial markets. These relationships are particularly evident in the following aspects:

(1) Market Participation:

① Key Factors: TURN (trading volume), NIA (number of new accounts).

② Trading volume and the number of newly opened accounts serve as indicators of investor enthusiasm. When market sentiment strengthens, these metrics tend to rise, and S^{GAN} correspondingly reflects this upward trend.

(2) Speculative Trading

① Key Factors: IPOR (IPO first-day return), IPON (number of IPOs).

② A heightened preference for high-risk assets is often accompanied by an increase in IPO first-day returns and the number of IPOs. S^{GAN} captures this dynamic, signaling periods of intensified speculative activity.

(3) Market Uncertainty

① Key Factor: EPU (economic policy uncertainty index).

② During times of rising uncertainty—such as policy shifts or geopolitical tensions—investor sentiment tends to weaken. This is reflected in a decline in S^{GAN}, indicating a more cautious market outlook.

5.3. Predictability of S^{GAN} in the American Stock Market

In order to briefly examine the dimensionality reduction effect of the GANs on various investor sentiment proxies in different market environments, we selected the five most representative sentiment proxies in the U.S. stock market for dimensionality reduction analysis. These five indicators are dividend premium (PDND), closed-end fund discount (CEFD), number of IPOs (NIPO), first-day returns on IPOs (RIPO), and equity share in new issues (S). They are all from the New York University Stern School of Business, while excess market returns (R^m) data are from the Tuck School of Business at Dartmouth. The period selected for this test is from November 2003 to November 2023.

Based on the similar methodology in Section 4.1, our empirical analysis shows that investor sentiment has a significant positive impact on stock market returns. Figure 13 compares the predicted returns with actual returns. As can be seen from the figure, the overall trend of actual returns is closer to that of predicted returns, suggesting that the model has demonstrated some validity in capturing the overall direction and trend of market returns. In addition, the F-statistic of the regression analysis and its corresponding p-value indicate that the overall model is highly significant, which further validates that the downgraded sentiment proxies are still effective in capturing the dynamic relationship of market returns. Moreover, we use the Diebold–Mariano test to assess the statistically significant differences in the forecasting models. The results show that S^{GAN} has a significantly superior forecasting accuracy with the DM statistic of 2.87, and the p-value is less than 0.01. It confirms the robustness of our model's predictive ability in different market

environments. This formal statistical test provides an objective basis for this model. In other words, the investor sentiment extracted by the GAN method still represents excellent explanatory ability in the prediction of U.S. market returns.

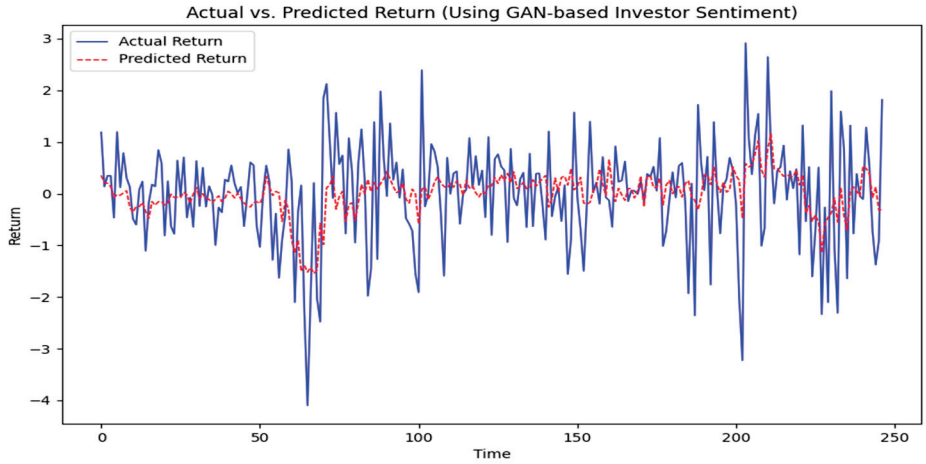


Figure 13. Comparison of the predicted returns and actual returns in U.S. market.

5.4. Theoretical Basis and Mechanism Analysis of S^{GAN}'s Superiority

This section explores the intrinsic theoretical basis of the superiority of S^{GAN} over traditional methods. The results of the empirical analyses show that S^{GAN} significantly outperforms methods such as S^{PCA}, S^{PLS}, and S^{GA} in predicting the excess returns of China's A-share market, and this superiority stems from the following key mechanisms.

Firstly, the nonlinear feature extraction capability of GANs enables S^{GAN} to effectively capture the complex relationships among sentiment proxy variables. Financial markets are inherently nonlinear systems, and the relationship between investor sentiment and market behavior often exhibits complex nonlinear patterns. Traditional methods such as PCA and PLS are mainly based on linear assumptions, and it is difficult for them to capture this complexity. GANs, on the other hand, are able to automatically learn the nonlinear dependencies between sentiment agents through multilayer neural networks and adversarial training without the need to pre-specify the form of the parameters. The feature importance analysis in Figure 9 confirms the more complex and balanced emotional structure captured by S^{GAN}.

Secondly, the adversarial training mechanism provides powerful noise filtering capabilities. Financial market data typically contain a large amount of noise that may interfere with the extraction of sentiment signals. The discriminator of S^{GAN} effectively filters out extraneous noise by constantly challenging the generator and forcing it to extract the most informative sentiment factors. This mechanism is particularly important during periods of high market volatility, explaining why S^{GAN} performed stably during the 2008 financial crisis and the 2015 market turmoil.

Thirdly, S^{GAN}'s adaptive feature weight assignment mechanism makes it highly sensitive to changes in the market environment. Unlike the static weights used by S^{PCA} and S^{PLS}, S^{GAN} is able to dynamically adjust feature weights according to the market environment. This feature is particularly critical for capturing rapid changes in investor sentiment, such as at bull–bear transition points, where S^{GAN} can quickly adjust the weights of liquidity indicators such as TURN and RV to more accurately predict market turns. This explains why S^{GAN} significantly outperforms other indicators during bear markets, as shown in Table 10.

Finally, the integrated loss function we designed ensures that S^{GAN} maintains economic explanatory power while preserving statistical significance. The λ-sensitivity analy-

sis in Figure 5 shows that this balance is critical to model performance. In addition, the model collapse prevention strategy we employ enhances the generalization ability of S^{GAN} to maintain stable performance under different market conditions.

In summary, the theoretical basis of S^{GAN}'s superiority lies in its ability to capture complex nonlinear sentiment structures, filter market noise, dynamically adapt to changes in market environments, and strike a balance between statistical significance and economic explanatory power through the unique architecture of GANs. These properties make S^{GAN} promising for a wide range of applications in asset pricing and market forecasting.

5.5. Computational Complexity and Limitations of Proposed Approach

In this test, the GANs need to compute forward propagation and backpropagation once for both the generator and discriminator within each epoch. Assuming that the dimension of our input data is d, and we have trained E epochs, the total computational complexity of each epoch is $o(E * n * (d + 32))$, which is the computational complexity of each iteration in the training process. The total training time is 1.53 s. However, the GAN method in this experiment has some limitations. Firstly, the GANs in this study use a single potential dimension to represent investor sentiment. Nevertheless, financial market sentiment is complex, and multiple dimensions may be required to adequately describe investor mood swings and market psychology. If the potential dimension is too small, it may not capture all the useful sentiment signals; if it is too large, it may introduce noise or cause excessive model complexity. Second, for the agents that need to be downgraded in this study, we selected appropriate hyper-parameters to enable effective cooperation between the generator and the discriminator, so that better quality sentiment signals can be extracted. However, if this method is extended without improvement to deal with complex financial data problems, the gaming process between generators and discriminators may lead to unstable training, which makes the extracted correlation signals distorted. Meanwhile, GANs are highly prone to overfitting the training data when dealing with correlated data. Especially in financial data, the market data have strong time dependence and seasonal fluctuations, and overfitting may lead to an inconsistent performance of sentiment signals in different time periods. Finally, financial market sentiment is determined not only by factors such as historical prices and technical indicators, but is also influenced by external factors such as macroeconomics, policies, international events, and so on. The GAN model may not be able to adequately capture the impact of these factors on sentiment, resulting in the extracted sentiment signals failing to reflect the complete market psychology.

Regarding the applicability of S^{GAN} in different trading frequency scenarios, although this study's analyses are mainly based on monthly data, the method has flexibility and limitations in practical application. For high-frequency trading environments, S^{GAN} faces two main challenges, which are computational latency and the high-frequency nature of emotional agents. The training process of GAN models is relatively time-consuming and is not suitable for real-time re-training in very high-frequency trading with millisecond decision requirements. However, the inference phase of the trained models is very efficient, which means that S^{GAN} can be useful in low- to medium-frequency trading strategies, such as intraday trading and daily or weekly investment decisions. Hybrid strategies can be used in practical applications: for example, the model can be updated weekly using the latest data, while the prediction process can take place in real time. In addition, high-frequency application scenarios require the redesign of sentiment proxies to suit short-term market volatility, which may be an important direction for future research. Overall, S^{GAN} is best suited for low- to medium-frequency trading strategies with daily to monthly time horizons and is able to effectively balance predictive performance and computational efficiency in these scenarios. For portfolio managers, S^{GAN} can be used as

an asset allocation decision tool to enable countercyclical investment strategies; market analysts can use it as a leading indicator of market turnaround, which performs particularly well in extreme market environments; and financial economists can use it to delve into the nonlinear relationship between sentiment and asset pricing.

6. Conclusions

Investor sentiment, a key behavioral indicator in the stock market, was constructed using various methods in prior research. This study introduces a novel investor sentiment, S^{GAN}, developed using GANs. Leveraging GANs, we synthesized latent factors from eight sentiment indicators and assessed their predictive abilities for excess returns in the Chinese A-share market. The empirical results demonstrate that GAN-based investor sentiment outperforms traditional sentiment measures derived from PCA, PLS, and GA in both in-sample tests and out-of-sample tests. Meanwhile, S^{GAN} remains statistically significant even after controlling for macroeconomic variables, indicating that it captures market-relevant information beyond economic fundamentals. Additionally, S^{GAN} exhibits strong predictive power across various industries, with the financial sector showing the highest sensitivity to investor sentiment, aligning with prior research findings. Last but not least, its predictive ability remains robust across different economic periods and countries, further affirming its stability.

In summary, this study innovatively applies GANs to investor sentiment modeling, breaking through the limitations of traditional linear downscaling methods and providing a new research paradigm for financial behavior. The unique ability of GANs to capture the nonlinear features of sentiment time series provides a new perspective to understand how market sentiment drives price fluctuations, and this methodological innovation not only improves the prediction accuracy, but also provides a theoretical basis for constructing more effective investment strategies. Moreover, we find that the lagged effects and asymmetric responses of sentiment indicators play a crucial part in the performance of GAN models. GANs are able to capture these lagged sentiment fluctuations efficiently through adversarial training to improve the predictive power of sentiment indicators, and at the same time deal with such asymmetric effects efficiently, so that the sentiment indicators can accurately reflect the way the market responds to different changes in sentiment. This enables GANs to outperform traditional approaches to sentiment analysis, providing more stable and predictive sentiment signals across a wide range of economic conditions, thereby improving the accuracy of the predictions of future market returns. Future research could be directed toward a comprehensive empirical investigation to select additional sentiment proxies and apply a GAN to develop more robust investor sentiment indicators. Additionally, it is of interest to adopt increasingly advanced dimensionality reduction methods to explore a broader range of stock markets.

Author Contributions: Conceptualization, S.Q.; methodology, S.Q. and Y.W.; software, S.Q.; validation, S.Q., Q.S. and Y.W.; formal analysis, S.Q., Q.S. and R.Z.; data curation, Y.W., Z.K., Z.L., R.Z. and K.O.; supervision, S.Q., Y.W. and Q.S.; writing—original draft preparation, S.Q.; writing—review and editing, Z.K. and K.O.; visualization, S.Q. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was conducted without any funding support.

Data Availability Statement: All data are included in the article.

Acknowledgments: The authors want to thank the editor and anonymous reviewers for their valuable suggestions for improving this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yang, C.; Zhou, L. Investor trading behavior, investor sentiment and asset prices. *N. Am. J. Econ. Financ.* **2015**, *34*, 42–62. [CrossRef]
2. Chen, C.; Liu, L.; Zhao, N. Fear Sentiment, Uncertainty, and Bitcoin Price Dynamics: The Case of COVID-19. *Emerg. Mark. Financ. Trade* **2020**, *56*, 2298–2309. [CrossRef]
3. Liu, Q.; Wang, X.; Du, Y. The weekly cycle of investor sentiment and the holiday effect—An empirical study of Chinese stock market based on natural language processing. *Heliyon* **2022**, *8*, e12646. [CrossRef] [PubMed]
4. Lu, J.; Chen, J. Asymmetric Relationship between Extreme Investor Sentiment and Stock Index. *Syst. Eng.* **2013**, *31*, 13–22.
5. Papapostolou, N.C.; Pouliasis, P.K.; Nomikos, N.K.; Kyriakou, I. Shipping investor sentiment and international stock return predictability. *Transp. Res. Part E Logist. Transp. Rev.* **2016**, *96*, 81–94. [CrossRef]
6. Hui, B.; Zheng, X.; Jia-Hong, L.; Junjie, W. Investor sentiment extracted from internet stock message boards and its effect on Chinese stock market. *J. Manag. Sci. China* **2018**, *21*, 86–101.
7. Hao, Y.; Chu, H.H.; Ko, K.C.; Lin, L. Momentum strategies and investor sentiment in the REIT market. *Int. Rev. Financ.* **2015**, *16*, 41–71. [CrossRef]
8. Dong, H.; Gil-Bazo, J. Sentiment stocks. *Int. Rev. Financ. Anal.* **2020**, *72*, 101573. [CrossRef]
9. Da, Z.; Engelberg, J.; Gao, P. The sum of all FEARS investor sentiment and asset prices. *Rev. Financ. Stud.* **2014**, *28*, 1–32. [CrossRef]
10. Bai, Y. Cross-border sentiment: An empirical analysis on EU stock markets. *Appl. Financ. Econ.* **2014**, *24*, 259–290. [CrossRef]
11. Chen, H.; Chong, T.T.L.; She, Y. A principal component approach to measuring investor sentiment in China. *Quant. Financ.* **2014**, *14*, 573–579. [CrossRef]
12. Chen, J.; Tang, G.; Yao, J.; Zhou, G. Investor attention and stock returns. *J. Financ. Quant. Anal.* **2021**, *57*, 455–484. [CrossRef]
13. Kelly, B.; Pruitt, S. Market expectations in the cross-section of present values. *J. Financ.* **2013**, *68*, 1721–1756. [CrossRef]
14. Huang, D.; Jiang, F.; Li, K.; Tong, G.; Zhou, G. Scaled PCA: A new approach to dimension reduction. *Manag. Sci.* **2022**, *68*, 1678–1695. [CrossRef]
15. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
16. Cheng, J.; Yang, Y.; Tang, X.; Xiong, N.; Zhang, Y.; Lei, F. Generative Adversarial Networks: A Literature Review. *Ksii Trans. Internet Inf. Syst.* **2020**, *14*, 4625–4647.
17. Kumar, A.; Alsadoon, A.; Prasad, P.W.C.; Abdullah, S.; Rashid, T.A.; Pham, D.T.H.; Nguyen, T.Q.V. Generative adversarial network (GAN) and enhanced root mean square error (ERMSE): Deep learning for stock price movement prediction. *Multimed. Tools Appl.* **2022**, *81*, 3995–4013. [CrossRef]
18. Nejad, F.S.; Ebadzadeh, M.M. Stock market forecasting using DRAGAN and feature matching. *Expert Syst. Appl.* **2024**, *244*, 122952. [CrossRef]
19. Wu, J.L.; Tang, X.R.; Hsu, C.H. A prediction model of stock market trading actions using generative adversarial network and piecewise linear representation approaches. *Soft Comput.* **2023**, *27*, 8209–8222. [CrossRef]
20. Xu, H.; Cao, D.; Li, S. A self-regulated generative adversarial network for stock price movement prediction based on the historical price and tweets. *Knowl.-Based Syst.* **2022**, *247*, 108712. [CrossRef]
21. Yan, D.; Li, B. Research on Stock Prediction Based on Generative Adversarial Networks. *Comput. Eng. Appl.* **2022**, *58*, 185–194.
22. Ma, Y.Y.; Liu, Y.Z.; Liu, C.L.; Zhang, T.J. Chinese Investors Multi-perspective Sentiment Analysis and Its Role in Stock Market Forecasting. *J. Northeast. Univ. (Nat. Sci.)* **2022**, *43*, 1201.
23. Zhang, W.; Gong, X.; Wang, C.; Ye, X. Predicting stock market volatility based on textual sentiment: A nonlinear analysis. *J. Forecast.* **2021**, *40*, 1479–1500. [CrossRef]
24. Shi, Y.; Tang, Y.R.; Long, W. Sentiment contagion analysis of interacting investors: Evidence from China's stock forum. *Phys. A Stat. Mech. Its Appl.* **2019**, *523*, 246–259. [CrossRef]
25. Trent, M.; Seale, H.; Chughtai, A.A.; Salmon, D.; MacIntyre, C.R. Trust in government, intention to vaccinate and COVID-19 vaccine hesitancy: A comparative survey of five large cities in the United States, United Kingdom, and Australia. *Vaccine* **2022**, *40*, 2498–2505. [CrossRef] [PubMed]
26. Gao, C.; Martin, I.W.R. Volatility, valuation ratios, and bubbles: An empirical measure of market sentiment. *J. Financ.* **2021**, *76*, 3211–3254. [CrossRef]
27. Davis, S.J.; Liu, D.; Sheng, X.S. Economic policy uncertainty in China since 1949: The view from mainland newspapers. *Work. Pap.* **2019**, *19*, 1–37.
28. Qadan, M.; Aharon, D.Y. Can investor sentiment predict the size premium? *Int. Rev. Financ. Anal.* **2019**, *63*, 10–26. [CrossRef]
29. Qadan, M.; Nama, H. Investor sentiment and the price of oil. *Energy Econ.* **2018**, *69*, 42–58. [CrossRef]
30. Slowik, A.; Kwasnicka, H. Evolutionary algorithms and their applications to engineering problems. *Neural Comput. Appl.* **2020**, *32*, 12363–12379. [CrossRef]

31. Pan, S.; Zheng, Z.; Guo, Z.; Luo, H. An optimized XGBoost method for predicting reservoir porosity using petrophysical logs. *J. Pet. Sci. Eng.* **2022**, *208*, 109520. [CrossRef]
32. Huang, D.; Jiang, F.; Tu, J.; Zhou, G. Investor sentiment aligned: A powerful predictor of stock returns. *Rev. Financ. Stud.* **2014**, *28*, 791–837. [CrossRef]
33. Neely, C.J.; Rapach, D.E.; Tu, J.; Zhou, G. Forecasting the Equity Risk Premium: The Role of Technical Indicators. *Manag. Sci.* **2014**, *60*, 1772–1791. [CrossRef]
34. Hou, Q.; Huo, X.; Leng, J.; Mannering, F. A note on out-of-sample prediction, marginal effects computations, and temporal testing with random parameters crash-injury severity models. *Anal. Methods Accid. Res.* **2022**, *33*, 100191. [CrossRef]
35. Wang, C.; Yang, J.; Zhang, B. A fault diagnosis method using improved prototypical network and weighting similarity-Manhattan distance with insufficient noisy data. *Measurement* **2024**, *226*, 114171. [CrossRef]
36. Song, Z.; Gong, X.; Zhang, C.; Yu, C. Investor sentiment based on scaled PCA method: A powerful predictor of realized volatility in the Chinese stock market. *Int. Rev. Econ. Financ.* **2023**, *83*, 528–545. [CrossRef]
37. Song, Z.Y.; Yu, C.R. Investor sentiment indices based on k-step PLS algorithm: A group of powerful predictors of stock market returns. *Int. Rev. Financ. Anal.* **2022**, *83*, 102321. [CrossRef]
38. Chen, Y.T.; Vincent, K. The Role of Momentum, Sentiment, and Economic Fundamentals in Forecasting Bear Stock Market. *J. Forecast.* **2016**, *35*, 504–527. [CrossRef]
39. Liang, C.; Xu, Y.A.; Wang, J.Q.; Yang, M. Whether dimensionality reduction techniques can improve the ability of sentiment proxies to predict stock market returns. *Int. Rev. Financ. Anal.* **2022**, *82*, 102169. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

mathematics

MDPI

Article

Contextual Analysis of Financial Time Series

Nadezhda Yarushkina, Aleksey Filippov * and Anton Romanov

Department of Information Systems, Ulyanovsk State Technical University, 32 Severny Venetz Street, 432027 Ulyanovsk, Russia; jng@ulstu.ru (N.Y.); a.romanov@ulstu.ru (A.R.)
* Correspondence: al.filippov@ulstu.ru; Tel.: +7-908-485-8390

Abstract: The evaluation of the financial state of small and medium-sized companies is a pressing issue today. This article introduces a novel method to evaluate a company's financial state, implemented as a module within a decision support system. This component uses fuzzy logic and knowledge engineering methods. The article describes an ontological model that provides the framework for data analysis and financial time-series modeling. The ontological framework simplifies the representation of the trends in the financial indicators under analysis. Integrating an ontology and a set of fuzzy rules makes it possible to develop control systems based on fuzzy inference. This approach provides the analysis and interpretation of results. The experimental results validate the accuracy and effectiveness of the proposed method.

Keywords: financial time series; contextual analysis; knowledge base; fuzzy inference

MSC: 37M10; 68T30

1. Introduction

This article describes an approach to the analysis of financial time series as a component of a decision support system to evaluate a company's financial state [1]. The primary users of this development are small and medium-sized companies. Precise analysis and prediction of trends in financial indicators are crucial for these companies for various reasons:

- The need to minimize errors in management decisions.
- The efficiency of resource allocation must be improved, as small companies typically have limited resources.
- Access to expensive consulting services for rapid analysis is often not applicable for small and medium-sized companies.
- Modern statistical software for operational analysis and forecasting of financial time series can be costly. In addition, these tools require managers to have expertise in mathematical statistics and significant intellectual and time investments. The existing models and methods for time-series analysis are complex and varied.

Consequently, small and medium-sized companies need affordable and user-friendly software solutions to evaluate their financial state, particularly in an unstable economic environment.

The application of machine learning methods in this context is challenging, as it complicates the understanding of the analysis and forecasting processes. Moreover, small companies typically lack the data needed to effectively use these methods [2].

2. Related Works

The analysis of financial indicators is crucial to the management of different systems [3,4]. This analysis can be conducted by examining risks and various factors that affect a company's sustainable growth and development.

Different approaches can be used for this analysis. Time-series analysis (both statistical and intelligent) is one of the most prevalent methods [5]. It is necessary to consider the historical context of these indicators [4,5]. Management decisions can then be applied to the company's financial system and broader management challenges.

The financial sector faces a variety of tasks, including identifying security issues [6,7], performing financial audits [8], and managing human resources [9].

Payment data from various systems currently show significant inconsistencies. These include payment processing data, transaction settlement data, and data from various e-commerce platforms that use different payment methods [10,11].

Various financial planning applications exist [12] that allow users to create financial plans based on personal data [13]. This trend originated from the needs of insurance companies [14–16], which developed an integrated knowledge-based model for AI-based advisors [17–20] to evaluate stock portfolios and analyze positive and negative responses to portfolio risk management. In addition, some applications are used to evaluate mutual fund services [21].

The following methods are commonly used for financial data analysis:

- Neural networks. The study in [8] discusses using an autoencoder to identify unusual entries in financial transaction logs. This method is helpful when dealing with datasets that have various features, making it challenging to establish simple statistical rules.
- Fuzzy logic methods. These methods demonstrate performance that is comparable to, or slightly less than, neural network models in terms of accuracy. However, they are excellent in explainability. The authors of [22] recommend fuzzy logic as a suitable approach for financial service applications.
- Ensemble learning methods. Modern approaches frequently use decision trees with gradient boosting. Researchers highlight the importance of feature selection for machine learning models, as it increases accuracy and improves the interpretability of results [23].

It is important to analyze not only the company's indicators in financial data analysis, but also to consider the impact of external factors on internal company processes. For example, several studies have investigated the effects of COVID-19 on company performance [24–26].

The primary focus of our research is to develop an approach that offers a meaningful interpretation of results with time-series analysis, fuzzy time-series modeling, and knowledge engineering methods. This proposed approach should be adaptable for small and medium-sized companies that may lack datasets applicable to machine learning methods. The innovation of this approach lies in the hybridization of time-series modeling and knowledge engineering for forecasting financial indicators and evaluating a company's financial state.

3. Financial Time-Series Analysis

Various behavioral patterns of objects over specific time intervals can be used as indicators to evaluate the state of a system. In addition, the preservation of long-term trends at a particular value of a significant indicator may signal a dangerous situation.

Time-series analysis involves the construction of a model that describes this series. The resulting model can identify dependencies and be used for forecasting. A critical aspect of the analysis is capturing the semantic component of the time series.

In financial time-series analysis, the semantic interpretation of forecast trends is a key point. Significant attributes of the trend include its duration, intensity, and direction. Experts can express knowledge about technical and financial time series with statements like: "If there is a long-term intensive increase in the value of a certain indicator, then ...", "If there is a short, slight decline in the value of a certain indicator, then ...", and "If a certain indicator shows long-term stability, then ...", etc. This raises the challenge of describing the time series in these specified terms.

The formal representation of the time series can be expressed as

$$TS = \{ts_i\}, i \in N,$$

where $ts_i = [t_i, v_{t_i}]$ indicates an element of the time series at time t_i, with v_{t_i} being the corresponding value.

The proposed approach includes the following steps:

1. Determine the piecewise linear trend of the time series through smoothing [27]:

$$Tend_{TS} = Tend(TS), Tend_{TS} = tend_j, j < i.$$

The smoothed time series can be determined as

$$ts'_t = k * ts_t + (1 - k) * (ts'_{t-1} - T_{t-1}),$$

where ts'_t represents the smoothed value at the current time t; k is the time-series-smoothing coefficient; ts_t is the current value of the series; ts'_{t-1} denotes the smoothed value from the previous period; $T_{t-1} = b * (ts'_t - ts'_{t-1}) + (1 - b) * T_{t-1}$ indicates the trend value from the previous period; and b is the trend-smoothing coefficient.
The coefficients k and b were selected through a grid search aimed at maximizing forecast accuracy on the test interval of the time series.

2. Label the levels of trends in the time series with semantically meaningful descriptions. The expert creates the set of labels according to the task requirements. A fuzzy representation of the trend series can be expressed as

$$Tend'_{TS} = Fuzzy(Tend_{TS}).$$

3. Predict the following value of the trend series:

$$tend'_{j+1} = Forecast(Tend_{TS}),$$

where $j + 1$ denotes the time point for the trend forecast.

4. Provide a semantic interpretation of the identified trend. Create the forecast for the time series for the upcoming period based on the inference using a knowledge base. Section 4 contains a complete description of this process.

4. Contextual Analysis of Financial Time Series

Our research focuses on describing and predicting a company's financial state through the analysis of time-series data. This task includes the following subtasks:

- Identify the indicators that define a company's financial state. Table 1 presents these indicators. It is also necessary to determine the methodology for the calculation

of indicator values and trends and the criteria to evaluate a company's financial state. The calculation formulas are derived from the values of the regulated financial statements in the Russian Federation: the balance sheet (form 1) and the profit and loss statement (form 2).

- Construct time series for each selected indicator and identify the general trends for each series. This process typically involves extracting and accumulating data from the company's information system databases. Therefore, it is necessary to analyze the company's statements over multiple periods and recalculate the indicators using the established formulas.
- Evaluate the financial state of the company using a set of expert rules adapted to the chosen indicators. Each indicator has unique characteristics regarding its dynamics and standard value ranges. For example, an increase in the current liquidity ratio positively influences the company's financial state, while a decrease adversely affects it. The current liquidity ratio being between 1 and 2 is normal; a value below 1 shows that a company lacks sufficient working capital to meet its short-term debt obligations, whereas a ratio above 2 can also negatively impact the company's financial state, suggesting inefficient use of funds. The company's financial state may be evaluated according to these guidelines.
- Analyze the trend and forecast future trends for each indicator, providing an economic interpretation of these projections. Understanding the expected dynamics of the indicators is crucial for correct management decisions.

  The input data include the time series of the selected financial indicators, while the output data include the predicted trends of these indicators, along with their contextual interpretations.

Table 1. Indicators of a company's financial state [28–32].

Indicator	Calculation Formula
Current ratio	Current assets/Current liabilities
Quick ratio	(Current assets-Reserve)/Current liabilities
Indebt ratio	Equity/Assets
Cash ratio	Equity/Liabilities
Reserve ratio	Own working capital/Reserve
Capitalization ratio	Attract capital/Own working capital

Proposed Rule Base for Financial State Evaluation

The financial state model Φ enables the evaluation of a company's financial state based on its indicators [28–32]. This model simplifies both predictive and descriptive analytics. A collection of financial indicators characterizes any company, and these indicators can be extracted from the company's information systems. Some indicators can be calculated from the values of other indicators. The system of indicators used to evaluate a company's financial state can be conceptualized as a tree structure. This relationship uses $\Phi(I)$ to denote an indicator I within the proposed model.

The dependencies between the model indicators can be represented as $\mathbb{D}$. The function $\uparrow$ allows for the collection of a set of indicators necessary for calculating the current indicator, provided that such dependencies exist. The expression $I^{\mathbb{D}} = \uparrow {}^{l} I$ shows that I is a derived indicator, and $I^{\mathbb{D}}$ includes the values required for its calculation. If two indicators I and J belong to $\mathbb{D}$, it can be said that I depends on J, and represented as $I \Rightarrow J$, with $I^{\mathbb{D}} = \uparrow {}^{1} I$. In addition, transitive dependencies may exist for some indicators, such as $I \Rightarrow \ldots \Rightarrow J \Rightarrow \ldots \Rightarrow K$, where $I^{\mathbb{D}} = \uparrow {}^{l} I$, and $l \geq 2$.

The rules regulating the calculation of the indicators are defined by the set $\mathbb{R}$. Each rule specifies a method for computing an indicator I by considering the dependencies $I^{\mathbb{D}} =\uparrow {}^{1}I$, using arithmetic operations from the set $\mathbb{F} = \langle +, -, \times, \div \rangle$.

The following triplet can represent the rule for calculating the indicators:

$$r = \langle I, \mathbb{A}, I^{\mathbb{D}} \rangle, I^{\mathbb{D}} =\uparrow {}^{1}I, \tag{1}$$

where $I \in \Phi$ is an indicator for calculating the company's financial state; $\mathbb{A} = \{a_1, a_2, \ldots, a_n\}$ represents the set of rules s; and $I^{\mathbb{D}}$ describes the set of indicators required to calculate the indicator I.

Each atom of the calculation rule can be described as a triplet:

$$a^r = \langle I_i^{\mathbb{D}}, \mathbb{F}_j, I_k^{\mathbb{D}} \rangle,$$

where $I_i^{\mathbb{D}}$, $I_k^{\mathbb{D}} \in I^{\mathbb{D}}$, $I^{\mathbb{D}} =\uparrow {}^{1}I$, $I \Rightarrow I_i^{\mathbb{D}}$, $I \Rightarrow I_k^{\mathbb{D}}$ represents the indicators for calculating the value of the indicator I; and $\mathbb{F}_j \in \mathbb{F}$ is an arithmetic operation.

The dependency tree built from $\mathbb{D}$ defines the order in which the rules are executed. It also considers transitive dependencies among indicators. The calculation of the financial state cannot proceed if any rule lacks the dependencies.

A set of rules $\mathbb{S}$ is used to evaluate a company's financial state. Each rule within $\mathbb{S}$ specifies the condition of a particular indicator I. Additionally, dependencies can be established between indicators to further evaluate the company's financial state. The structure of a rule $s \in \mathbb{S}$ resembles that of a rule $r \in \mathbb{R}$ (as shown in expression (1)). The distinction between the rules s and r lies in the composition of the atoms in the s rule:

$$a^s = \langle I_i^{\mathbb{D}}, and, I_k^{\mathbb{D}} \rangle,$$

where the logical operator *and* is used instead of the arithmetic operation. The logical operator *or* is implemented by creating additional rules:

$$s_1 \; or \; s_2 \; or \; \ldots \; or \; s_n.$$

Let us explore an example of the proposed methodology. Semantic Web technologies have been selected to represent expert knowledge. The OWL 2 language [33] is used to represent the knowledge model, while the SWRL language [34] is used for rule definition. Inference based on the content of the knowledge base is produced using the Pellet reasoner [35], and the SWRLF reasoner [36] is applied for fuzzy logic inference.

Semantic Web technologies significantly reduce the costs associated with creating a knowledge base to evaluate a company's financial state. The combination of OWL 2, SWRL, and the Pellet reasoner simplifies the verification of ontology consistency and the generation of new knowledge through inference. Fuzzy inference allows experts to define the rules to evaluate the financial state in verbal terms, minimizing the costs related to rule formation.

The $\mathcal{SHOIQF}(\mathcal{D})$ (DL) [37] dialect of description logic is used to represent the knowledge model of the proposed knowledge base. Table 2 presents the DL operators and axioms used to represent the components of the proposed knowledge base model.

The *TBox* terminology of the proposed knowledge base includes a definition of the following classes:

- The Finance class describes a set of indicators to evaluate of the company's financial state;
- The Object class describes a company.

The Finance and Object classes are declared disjoint:

$$\mathsf{Finance} \sqcap \mathsf{Object} \sqsubseteq \bot.$$

An individual cannot belong to several classes at the same time if the classes are declared disjoint.

Let us consider the indicator classes as an element of the terminology of the proposed knowledge base:

- The StateIndicators class describes a set of indicators extracted from numerical data found in the company's balance sheet.
- The State class describes potential states of the company. The state is inferred by the StateIndicators indicator.

Table 2. DL operators and axioms [38].

Description	DL	OWL
Top (a special class with every individual as an instance)	$\top$	owl:Thing
Bottom (an empty class)	$\bot$	owl:Nothing
Class inclusion axiom	$A \sqsubseteq B$	A owl:SubClassOf B
Disjoint classes axiom	$A \sqcap B \sqsubseteq \bot$	[A, B] owl:DisjointClasses
Equivalence classes axiom (or defining classes with necessary and sufficient conditions)	$A \equiv B$	[A, B] owl:equivalentClasses
Intersection or conjunction of classes	$A \sqcap B$	A and B
Universal restriction axiom	$\forall R.A$	R only A
Existential restriction axiom	$\exists R.A$	R some A
Cardinality restrictions axiom	$\leq nR.A$	R exactly n A
Concept assertion axiom (a is an instance of class A)	$a: A$	a: A
Role assertion axiom	$(a, b): R$	a R b

The StateIndicators and State classes are declared as disjointed:

$$\mathsf{StateIndicators} \sqcap \mathsf{State} \sqsubseteq \bot.$$

The StateIndicators class has the following subclasses:

- CapitalizationRatio;
- CashRatio;
- CurrentRatio;
- IndebtRatio;
- QuickRatio;
- ReserveRatio.

The classes mentioned earlier relate to a set of indicators reflecting a company's financial state, as shown in Table 1. All classes used to characterize these financial indicators are defined as disjoint:

$$\mathsf{CapitalizationRatio} \sqcap \mathsf{CashRatio} \sqcap \mathsf{CurrentRatio} \sqcap$$
$$\sqcap \mathsf{IndebtRatio} \sqcap \mathsf{QuickRatio} \sqcap \mathsf{ReserveRatio} \sqsubseteq \bot.$$

Calculations must be conducted to determine the company's financial state indicators. The starting data for these calculations are the numerical data from the company's balance sheet. The following attributes of the Object class are used to evaluate the values of these numerical indicators:

$$
\begin{aligned}
\mathsf{Object} \sqsubseteq\ & \exists \mathsf{assets.Double} \sqcap \forall \mathsf{assets.Double} \sqcap \\
& \sqcap = 1\mathsf{assets.Double} \sqcap \\
& \sqcap \exists \mathsf{attractCapital.Double} \sqcap \forall \mathsf{attractCapital.Double} \sqcap \\
& \sqcap = 1\mathsf{attractCapital.Double} \sqcap \\
& \sqcap \exists \mathsf{currentAssets.Double} \sqcap \forall \mathsf{currentAssets.Double} \sqcap \\
& \sqcap = 1\mathsf{currentAssets.Double} \sqcap \\
& \sqcap \exists \mathsf{currentLiabilities.Double} \sqcap \forall \mathsf{currentLiabilities.Double} \sqcap \\
& \sqcap = 1\mathsf{currentLiabilities.Double} \sqcap \\
& \sqcap \exists \mathsf{equity.Double} \sqcap \forall \mathsf{equity.Double} \sqcap \\
& \sqcap = 1\mathsf{equity.Double} \sqcap \\
& \sqcap \exists \mathsf{liabilities.Double} \sqcap \forall \mathsf{liabilities.Double} \sqcap \\
& \sqcap = 1\mathsf{liabilities.Double} \sqcap \\
& \sqcap \exists \mathsf{ownWorkingCapital.Double} \sqcap \forall \mathsf{ownWorkingCapital.Double} \sqcap \\
& \sqcap = 1\mathsf{ownWorkingCapital.Double} \sqcap \\
& \sqcap \exists \mathsf{reserve.Double} \sqcap \forall \mathsf{reserve.Double} \sqcap \\
& \sqcap = 1\mathsf{reserve.Double}.
\end{aligned}
$$

Each property of the Object class is functional and corresponds to a company balance sheet indicator. Object class properties are used to calculate the indicators of a company's financial state (Table 1).

The expert formed an SWRL rule ($\in \mathbb{R}$) to calculate the value of the ownWorkingCapital property (indicator):

$$
\begin{aligned}
& \mathsf{currentAssets}(?o, ?ca) \wedge \mathsf{currentLiabilities}(?o, ?cl) \wedge \\
& \quad \wedge \mathsf{swrlb} : \mathsf{subtract}(?owc, ?ca, ?cl) \rightarrow \mathsf{ownWorkingCapital}(?o, ?owc)
\end{aligned} \tag{2}
$$

An SWRL rule consists of two parts: a condition (antecedent) and a conclusion (consequent). The antecedent and consequent of each SWRL rule contain a set of atoms. Each atom is related to an ontology axiom of $TBox$ or $ABox$, as well as to a set of variables. The atoms within the rule are joined by the logical operator AND ($\wedge$). The antecedent of Rule (2) is

$$
\mathsf{currentAssets}(?o, ?ca) \wedge \mathsf{currentLiabilities}(?o, ?cl) \wedge \mathsf{swrlb} : \mathsf{subtract}(?owc, ?ca, ?cl)
$$

The antecedent of Rule (2) contains the following atoms:

- $\mathsf{currentAssets}(?o, ?ca)$. This atom can be interpreted as: For any individual in the ontology that has the currentAssets property, retrieve the value of that property and assign it to the ?ca variable, while also saving a reference to the individual in the ?o variable.
- $\mathsf{currentLiabilities}(?o, ?cl)$. Comparable to the previous atom, but focused on the currentLiabilities property.
- $\mathsf{swrlb} : \mathsf{subtract}(?owc, ?ca, ?cl)$. The swrlb namespace comprises the implementation of arithmetic operations ($\mathbb{F}$). This rule atom computes the result of subtracting the variable ?cl from the variable ?ca, and the resulting value is assigned to the variable ?owc.

A list of values is generated as a result of executing the antecedent of Rule (2). Each item in this list can be expressed as

$$antecedent_i = \langle ?o, ?cl, ?ca, ?owc \rangle,$$

where each individual ?o is assigned the values ?ca and ?cl for the properties currentAssets and currentLiabilities, respectively, along with the result of the subtraction ?owc.

The consequent of Rule (2) contains only one atom:

- ownWorkingCapital(?o, ?owc). This atom should be interpreted as: Create a property called ownWorkingCapital for the individual ?o and assign the value of the variable ?owc to this property.

Therefore, Rule (2) describes the expression

$$\mathsf{ownWorkingCapital} = \mathsf{currentAssets} - \mathsf{currentLiabilities}.$$

Additionally, a set of functional properties has been defined for the Object class to represent the values of financial state indicators (Table 1):

$$\begin{aligned}
\mathsf{Object} \sqsubseteq\ & \exists \mathsf{capitalizationRatio.Double} \sqcap \forall \mathsf{capitalizationRatio.Double} \sqcap \\
& \sqcap = 1 \mathsf{capitalizationRatio.Double} \sqcap \\
& \sqcap \exists \mathsf{cashRatio.Double} \sqcap \forall \mathsf{cashRatio.Double} \sqcap \\
& \sqcap = 1 \mathsf{cashRatio.Double} \sqcap \\
& \sqcap \exists \mathsf{currentRatio.Double} \sqcap \forall \mathsf{currentRatio.Double} \sqcap \\
& \sqcap = 1 \mathsf{currentRatio.Double} \sqcap \\
& \sqcap \exists \mathsf{indebtRatio.Double} \sqcap \forall \mathsf{indebtRatio.Double} \sqcap \\
& \sqcap = 1 \mathsf{indebtRatio.Double} \sqcap \\
& \sqcap \exists \mathsf{quickRatio.Double} \sqcap \forall \mathsf{quickRatio.Double} \sqcap \\
& \sqcap = 1 \mathsf{quickRatio.Double} \sqcap \\
& \sqcap \exists \mathsf{reserveRatio.Double} \sqcap \forall \mathsf{reserveRatio.Double} \sqcap \\
& \sqcap = 1 \mathsf{reserveRatio.Double}.
\end{aligned}$$

Each of the functional properties mentioned above defines the set $\mathbb{D}$.

The values of a company's financial state indicators are calculated through an inference. The calculation is based on the following SWRL rules from the set $\mathbb{R}$ (Table 1):

$$\begin{aligned}
& \mathsf{attractCapital}(?o, ?ac) \wedge \mathsf{ownWorkingCapital}(?o, ?owc) \wedge \\
& \quad \wedge \mathsf{swrlb:divide}(?gr, ?ac, ?owc) \rightarrow \mathsf{capitalizationRatio}(?o, ?gr) \\
& \mathsf{equity}(?o, ?e) \wedge \mathsf{liabilities}(?o, ?l) \wedge \\
& \quad \wedge \mathsf{swrlb:divide}(?wcr, ?e, ?l) \rightarrow \mathsf{cashRatio}(?o, ?wcr) \\
& \mathsf{currentAssets}(?o, ?ca) \wedge \mathsf{currentLiabilities}(?o, ?cl) \wedge \\
& \quad \wedge \mathsf{swrlb:divide}(?cr, ?ca, ?cl) \rightarrow \mathsf{currentRatio}(?o, ?cr) \\
& \mathsf{equity}(?o, ?e) \wedge \mathsf{assets}(?o, ?a) \wedge \mathsf{swrlb:divide}(?fir, ?e, ?a) \rightarrow \mathsf{indebtRatio}(?o, ?fir) \\
& \mathsf{currentAssets}(?o, ?ca) \wedge \mathsf{currentLiabilities}(?o, ?cl) \wedge \mathsf{reserve}(?o, ?r) \wedge \\
& \quad \wedge \mathsf{swrlb:subtract}(?car, ?ca, ?r) \wedge \mathsf{swrlb:divide}(?qr, ?car, ?cl) \rightarrow \mathsf{quickRatio}(?o, ?qr) \\
& \mathsf{ownWorkingCapital}(?o, ?owc) \wedge \mathsf{reserve}(?o, ?r) \wedge \\
& \quad \wedge \mathsf{swrlb:divide}(?rr, ?owc, ?r) \rightarrow \mathsf{reserveRatio}(?o, ?rr).
\end{aligned}$$

The following properties of the Object class are used to evaluate a company's financial state by analyzing the values of the calculated indicators:

$$\begin{aligned}\mathsf{Object} \sqsubseteq\ & \forall \mathsf{hasCapitalizationRatio.CapitalizationRatio} \sqcap \\ & \sqcap \forall \mathsf{hasCashRatio.CashRatio} \sqcap \\ & \sqcap \forall \mathsf{hasCurrentRatio.CurrentRatio} \sqcap \\ & \sqcap \forall \mathsf{hasIndebtRatio.IndebtRatio} \sqcap \\ & \sqcap \forall \mathsf{hasQuickRatio.QuickRatio} \sqcap \\ & \sqcap \forall \mathsf{hasReserveRatio.ReserveRatio} \sqcap \\ & \sqcap \forall \mathsf{hasState.State}.\end{aligned}$$

The values of the properties mentioned above are calculated through fuzzy inference by the SWRLF module. The functions of the SWRLF module are based on OWL annotations, with each type of annotation setting the parameters and configurations of the fuzzy inference algorithm [36].

Fuzzy inference consists of the following steps [39,40]:

1. Fuzzification. This process involves translating numerical indicators of object properties into linguistic terms. Each value of the input variables is associated with a specific linguistic term, along with a corresponding membership value. The input variables for fuzzification are the numerical values of the financial indicators.
2. Aggregation. The truth degree for the antecedents of each rule from the $\mathbb{S}$ set is determined during the aggregation phase. This truth degree is calculated using the membership value associated with the linguistic term. The truth degree of an antecedent is equal to the truth degree of an atom if an antecedent of a fuzzy rule comprises a single atom. When an antecedent includes multiple atoms, the truth degree is calculated by applying fuzzy logic operations on the truth degrees of the antecedent atoms, typically using the fuzzy logical AND (min) operator.
3. Activation. The truth degree for each consequent atom of a fuzzy rule is determined at the activation stage. This degree is calculated as the algebraic product of the weight of the current rule and the truth degree of its antecedent. In addition to the algebraic product, minimum and average functions can also determine the truth degree. If the weight for a rule is not defined, it defaults to one.
4. Accumulation. The accumulation stage involves forming a membership function for each linguistic variable derived from the consequents of activated fuzzy rules. A rule is activated if the truth degree of its consequent is greater than zero. The accumulation process is based on the union of fuzzy sets belonging to all consequent atoms related to a particular linguistic variable.
5. Defuzzification. The result of defuzzification yields crisp values for each output linguistic variable, based on the accumulation results of all output linguistic terms from the fuzzy rule consequent. The set of output linguistic variables is determined based on the atoms of the consequent of the activated rules.

The stages of fuzzy inference can be executed through multiple methods. Various fuzzy logic algorithms, various techniques for accumulating sets, and differing strategies for activation and defuzzification can all be used.

The properties of the Object class listed below were labeled as input variables for the fuzzy inference algorithm using the fuzzyInputVariable annotation:

- capitalizationRatio;
- cashRatio;
- currentRatio;

- indebtRatio;
- quickRatio;
- reserveRatio.

Specific *ABox* axioms were used to describe linguistic terms for each indicator. For example, regarding the indicator *CapitalizationRatio* this is as follows:

$$\begin{aligned}
&\mathsf{CapitalizationRatioLow}: \mathsf{CapitalizationRatio}\\
&\mathsf{CapitalizationRatioMiddle}: \mathsf{CapitalizationRatio}\\
&\mathsf{CapitalizationRatioHigh}: \mathsf{CapitalizationRatio}\\
&\mathsf{CapitalizationRatio} \sqsubseteq \{\\
&\quad \mathsf{CapitalizationRatioLow}, \mathsf{CapitalizationRatioMiddle}, \mathsf{CapitalizationRatioHigh}\\
&\}.
\end{aligned}$$

The fuzzyTerm annotation was used to define the relationship between the input variable and its associated linguistic terms and membership functions. The settings for all linguistic terms are presented in Table 3.

Table 3. Fuzzy inference settings [28–32].

Fuzzy Term	Input Variable	Membership Function
CapitalizationRatioLow	capitalizationRatio	ZShape (0.33, 0.37)
CapitalizationRatioMiddle	capitalizationRatio	Trapezoid (0.35, 0.4, 0.6, 0.65)
CapitalizationRatioHigh	capitalizationRatio	SShape (0.63, 0.7)
CashRatioLow	cashRatio	ZShape (0.08, 0.15)
CashRatioMiddle	cashRatio	Trapezoid (0.1, 0.2, 0.8, 1.0)
CashRatioHigh	cashRatio	SShape (0.9, 1.1)
CurrentRatioLow	currentRatio	ZShape (0.4, 0.8)
CurrentRatioMiddle	currentRatio	Trapezoid (0.6, 1.0, 1.5, 1.9)
CurrentRatioHigh	currentRatio	SShape (1.6, 2.2)
IndebtRatioLow	indebtRatio	ZShape (0.35, 0.45)
IndebtRatioMiddle	indebtRatio	Trapezoid (0.4, 0.5, 0.6, 0.7)
IndebtRatioHigh	indebtRatio	SShape (0.65, 0.8)
QuickRatioLow	quickRatio	ZShape (0.42, 0.84)
QuickRatioMiddle	quickRatio	Trapezoid (0.83, 0.85, 0.87, 0.91)
QuickRatioHigh	quickRatio	SShape (0.9, 1.2)
ReserveRatioLow	reserveRatio	ZShape (0.3, 0.51)
ReserveRatioMiddle	reserveRatio	Trapezoid (0.47, 0.5, 0.57, 0.61)
ReserveRatioHigh	reserveRatio	SShape (0.6, 0.8)

5. Results

Here are some examples of fuzzy rules created by an expert to evaluate the current liquidity ratio (Figure 1):

1. Long-term growth indicates a positive situation for the company.
2. A long- or medium-term decrease indicates a negative situation for the company.
3. A short and intense decrease signifies that the company should focus more on its reserves and assets.

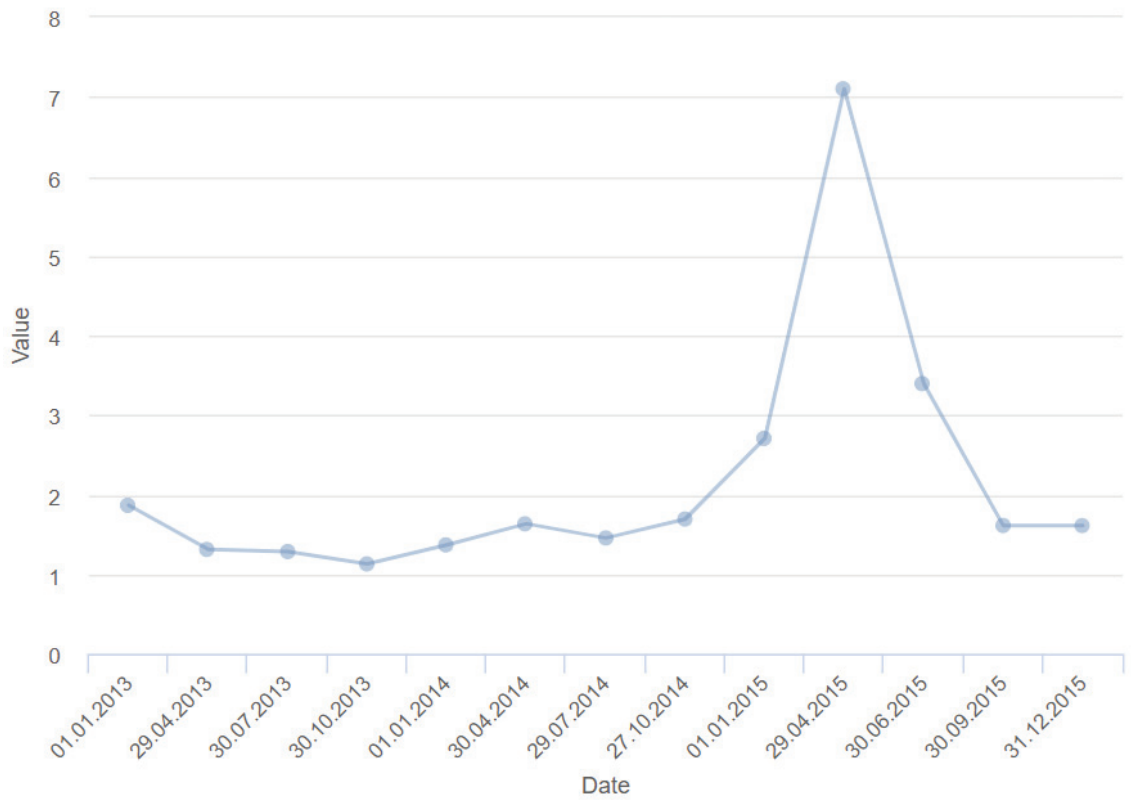


Figure 1. Current liquidity ratio.

The expert rules from the $\mathbb{S}$ set were formalized as the following SWRL rules [28–32]:

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioLow}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioLow}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateLow})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioMiddle}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioLow}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateLow})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioLow}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioMiddle}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateLow})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioMiddle}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioMiddle}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateMiddle})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioMiddle}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioHigh}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateMiddle})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioHigh}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioMiddle}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateMiddle})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioLow}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioHigh}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateLow})$$

$$\text{hasCurrentRatio}(?o, \text{CurrentRatioHigh}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioLow}) \rightarrow \\ \rightarrow \text{hasState}(?o, \text{StateMiddle})$$

$$\begin{aligned}
&\text{hasCurrentRatio}(?o, \text{CurrentRatioHigh}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioHigh}) \rightarrow \\
&\quad \rightarrow \text{hasState}(?o, \text{StateHigh}) \\
&\text{hasCurrentRatio}(?o, \text{CurrentRatioHigh}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioMiddle}) \wedge \\
&\quad \wedge \text{hasCapitalizationRatio}(?o, \text{CapitalizationRatioHigh}) \rightarrow \text{hasState}(?o, \text{StateHigh}) \\
&\text{hasCurrentRatio}(?o, \text{CurrentRatioMiddle}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioHigh}) \wedge \\
&\quad \wedge \text{hasCapitalizationRatio}(?o, \text{CapitalizationRatioHigh}) \rightarrow \text{hasState}(?o, \text{StateHigh}) \\
&\text{hasCurrentRatio}(?o, \text{CurrentRatioHigh}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioMiddle}) \wedge \\
&\quad \wedge \text{hasCashRatio}(?o, \text{CashRatioHigh}) \rightarrow \text{hasState}(?o, \text{StateHigh}) \\
&\text{hasCurrentRatio}(?o, \text{CurrentRatioMiddle}) \wedge \text{hasQuickRatio}(?o, \text{QuickRatioHigh}) \wedge \\
&\quad \wedge \text{hasCashRatio}(?o, \text{CashRatioHigh}) \rightarrow \text{hasState}(?o, \text{StateHigh}).
\end{aligned}$$

Let us take an example of evaluating a company's financial state. Define a specific company as an individual object of the Object class and set the values for its numerical indicators:

$$\begin{aligned}
&\text{object} : \text{Object} \\
&(\text{object}, 6154721) : \text{currentAssets} \\
&(\text{object}, 3818380) : \text{currentLiabilities} \\
&(\text{object}, 7018805) : \text{assets} \\
&(\text{object}, 7018800) : \text{liabilities} \\
&(\text{object}, 3818380) : \text{attractCapital} \\
&(\text{object}, 3622864) : \text{reserve} \\
&(\text{object}, 3200440) : \text{equity}.
\end{aligned}$$

The values of the properties related to the individual object were determined after applying the rules from the $\mathbb{R}$ set:

$$\begin{aligned}
&(\text{object}, 2336341) : \text{ownWorkingCapital} \\
&(\text{object}, 1.63) : \text{capitalizationRatio} \\
&(\text{object}, 0.46) : \text{cashRatio} \\
&(\text{object}, 1.61) : \text{currentRatio} \\
&(\text{object}, 0.46) : \text{indebtRatio} \\
&(\text{object}, 0.66) : \text{quickRatio} \\
&(\text{object}, 0.64) : \text{reserveRatio}.
\end{aligned}$$

The SWRL library has already implemented the $\uparrow$ function to control dependencies among rules and determine their execution order [41].

The results of the fuzzification stage for the defined input variables are presented in Table 4.

Table 4. Fuzzification results.

Fuzzy Term	Input Variable	Membership Degree
CurrentRatioLow	currentRatio	0.0
CurrentRatioMiddle	currentRatio	0.72
CurrentRatioHigh	currentRatio	0.00078
CapitalizationRatioLow	capitalizationRatio	0.0
CapitalizationRatioMiddle	capitalizationRatio	0.0
CapitalizationRatioHigh	capitalizationRatio	1.0
IndebtRatioLow	indebtRatio	0.0
IndebtRatioMiddle	indebtRatio	0.56
IndebtRatioHigh	indebtRatio	0.0
QuickRatioLow	quickRatio	0.65
QuickRatioMiddle	quickRatio	0.0
QuickRatioHigh	quickRatio	0.0
CashRatioLow	cashRatio	0.0
CashRatioMiddle	cashRatio	1.0
CashRatioHigh	cashRatio	0.0
ReserveRatioMiddle	reserveRatio	0.0
ReserveRatioLow	reserveRatio	1.0
ReserveRatioHigh	reserveRatio	0.1

Next, the steps of aggregation and activation were executed. Consequently, the truth degree of the consequent of the logical rules was calculated. The following rules have a truth degree value of the consequent greater than 0:

Truth degree: 0.65

$$\mathsf{hasCurrentRatio}(?o, \mathsf{CurrentRatioMiddle}) \wedge \mathsf{hasQuickRatio}(?o, \mathsf{QuickRatioLow}) \rightarrow \\ \rightarrow \mathsf{hasState}(?o, \mathsf{StateLow})$$

Truth degree: 0.00078

$$\mathsf{hasCurrentRatio}(?o, \mathsf{CurrentRatioHigh}) \wedge \mathsf{hasQuickRatio}(?o, \mathsf{QuickRatioLow}) \rightarrow \\ \rightarrow \mathsf{hasState}(?o, \mathsf{StateMiddle}).$$

Consequently, two conclusions can be inferred based on the company's state:

- StateLow with a confidence level of 0.65;
- StateMiddle with a confidence level of 0.00078.

The proposed approach for contextual evaluation of a company's financial state can compute various financial indicators. To achieve this, a set of rules must be defined to calculate these indicators. Each rule can describe the calculation method for a specific indicator based on the currently available data. A set of fuzzy rules must be created to evaluate a company's financial state. The crisp values of indicators that are not known at the moment can be calculated with fuzzy inference.

Figure 2 illustrates the time series forecast using the Holt exponential method with an additive trend, which does not include a seasonal component, compared to the forecast based on the knowledge base fuzzy inference (Figure 3).

An analysis of the numerical forecast might reveal that the "current liquidity ratio" indicator is stable. However, it is challenging to identify the economic impacts of random outliers or non-systematic factors given the historical fluctuations in its dynamics. Therefore, it is crucial to deal with expert evaluations and clarify the conclusions regarding the indicator forecast from the perspective of company management.

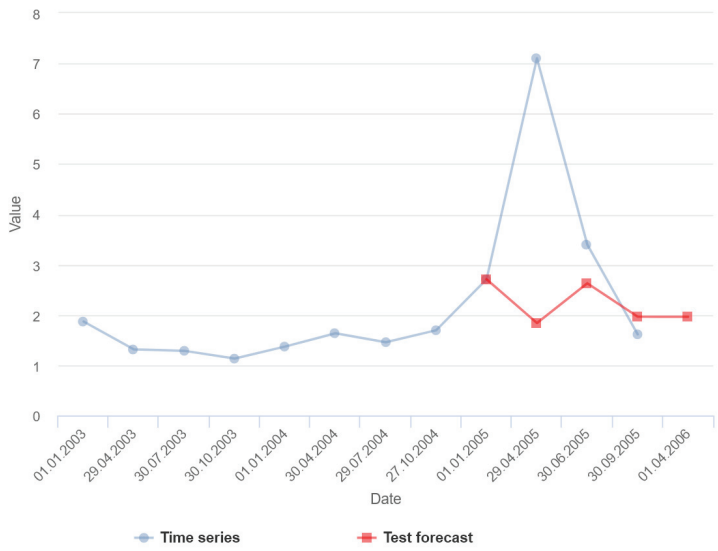


Figure 2. Current liquidity ratio forecast based on Holt method.

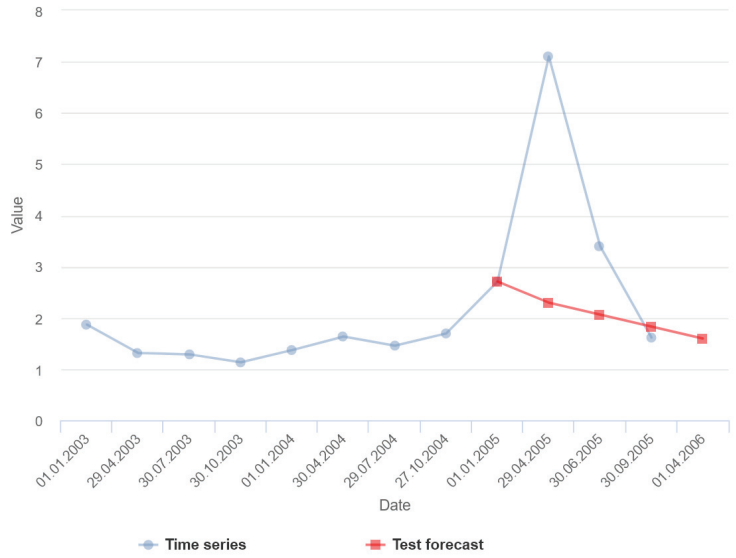


Figure 3. Current liquidity ratio rule-based forecast.

As shown in Figures 2 and 3, there can be notable differences in indicator evaluation during the model testing phase with the SMAPE criterion. The knowledge base creates forecasts that more accurately reflect the indicator behavior. Thus, the SMAPE estimation for the forecast shown in Figure 2 is 36%, and for Figure 3 it is 33%.

The following expression represents a balanced approach to forecasting:

$$forecast = \alpha * ts_forecast + \beta * kb_forecast, \tag{3}$$

where α and β are the weighting coefficients for the model components. In this context, $forecast$ indicates the overall trend in the forecast value, $ts_forecast$ reflects the trend derived from time-series analysis, and $kb_forecast$ represents the trend based on rule-based inference.

6. Conclusions

The proposed approach improves the quality of forecasting time series, particularly when there is a lack of enough data for analysis. Quantitative indicators that define a company's financial state represent data that are interesting for research. These indicators

describe a system and have obvious dependencies. The main aim of the study is the problem of predictive analytics, especially for small and medium-sized companies, which often have limited historical data on financial indicators. The two-component forecasting model developed in this research provides forecasts with higher quality compared to traditional methods. However, the limitations of this approach are the necessity of forming the rule base.

The results of ten experiments show that the proposed method achieves an average forecast quality of 12.8% according to the SMAPE criterion, compared to 16.2% for forecasts using exponential smoothing techniques. The forecasting task faced limitations for short time series, with an inability to extract periodic components, making it challenging to apply more advanced statistical and neural network models.

Author Contributions: Conceptualization, A.R.; data curation, A.F.; formal analysis, A.F.; methodology, N.Y.; visualization, A.R. All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: Ministry of Science and Higher Education of Russia: project No. 075-03-2023-143 331 "The study of intelligent predictive analytics based on the integration of methods for constructing features of heterogeneous dynamic data for machine learning and methods of predictive multimodal data analysis".

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Borisovna, I.T. Balance Sheet Analysis While Instructing on Financial Statements in English. *Balk. Sci. Rev.* **2020**, *4*, 2171.
2. Lee, I.; Shin, Y.J. Machine Learning for Enterprises: Applications, Algorithm Selection, and Challenges. *Bus. Horizons* **2020**, *63*, 157–170. [CrossRef]
3. Walter, C. Sustainable Financial Risk Modelling Fitting the SDGs: Some Reflections. *Sustainability* **2020**, *12*, 7789. [CrossRef]
4. Angelini, P. Financial Decisions Based on Zero-Sum Games: New Conceptual and Mathematical Outcomes. *Int. J. Financ. Stud.* **2024**, *12*, 56. [CrossRef]
5. Castillo, O.; Castro, J.R.; Melin, P. Interval Type-3 Fuzzy Aggregation of Neural Networks for Multiple Time Series Prediction: The Case of Financial Forecasting. *Axioms* **2022**, *11*, 251. [CrossRef]
6. Ratmono, D.; Darsono, D.; Cahyonowati, N. Financial Statement Fraud Detection With Beneish M-Score and Dechow F-Score Model: An Empirical Analysis of Fraud Pentagon Theory in Indonesia. *Int. J. Financ. Res.* **2020**, *11*, 154. [CrossRef]
7. Poltorak, A.; Potryvaieva, N.; Kuzoma, V.; Volosyuk, Y.; Bobrovska, N. Development of Doctrinal Model for State Financial Security Management and Forecasting Its Level. *East. Eur. J. Enterp. Technol.* **2021**, *5*, 26–33. [CrossRef]
8. Schultz, M.; Tropmann-Frick, M. Autoencoder Neural Networks Versus External Auditors: Detecting Unusual Journal Entries in Financial Statement Audits. In Proceedings of the 53rd Hawaii International Conference on System Sciences, Maui, HI, USA, 7–10 January 2020.
9. Saputra, K.A.K.; Subroto, B.; Rahman, A.F.; Saraswati, E. Financial Management Information System, Human Resource Competency and Financial Statement Accountability: A Case Study in Indonesia. *J. Asian Financ. Econ. Bus.* **2021**, *8*, 277–285. [CrossRef]
10. Bello, G.; Perez, A.J. Adapting Financial Technology Standards to Blockchain Platforms. In Proceedings of the 2019 ACM Southeast Conference, Kennesaw, GA, USA, 18–20 April 2019; pp. 109–116.
11. Moon, W.Y.; Kim, S.D. A Payment Mediation Platform for Heterogeneous Fintech Schemes. In Proceedings of the 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, Xi'an, China, 3–5 October 2016; pp. 511–516.
12. Lee, R.S.T. COSMOS Trader—Chaotic Neuro-Oscillatory Multiagent Financial Prediction and Trading System. *J. Financ. Data Sci.* **2019**, *5*, 61–82. [CrossRef]

13. Kumari, A.; Kumar Sharma, A. Infrastructure Financing and Development: A Bibliometric Review. *Int. J. Crit. Infrastruct. Prot.* **2017**, *16*, 49–65. [CrossRef]
14. Liu, Y.; Peng, J.; Yu, Z. Big Data Platform Architecture under the Background of Financial Technology—In the Insurance Industry as an Example. In Proceedings of the ACM International Conference 2018 on Big Data Engineering and Technology, Chengdu, China, 25–27 August 2018; pp. 31–35.
15. Stoeckli, E.; Dremel, C.; Uebernickel, F. Exploring Characteristics and Transformational Capabilities of InsurTech Innovations to Understand Insurance Value Creation in a Digital World. *Electron. Mark.* **2018**, *28*, 287–305. [CrossRef]
16. Marafie, Z.; Lin, K.J.; Zhai, Y.; Li, J. Proactive Fintech: Using Intelligent IoT to Deliver Positive Insurtech Feedback. In Proceedings of the 2018 20th IEEE International Conference on Business Informatics, Vienna, Austria, 11–14 July 2018; pp. 72–81.
17. Faloon, M.; Scherer, B. Individualization of Robo-Advice. *J. Wealth Manag.* **2017**, *20*, 30–36. [CrossRef]
18. Day, M.-Y.; Lin, J.-T.; Chen, Y.-C. Artificial Intelligence for Conversational Robo-Advisor. In Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, 28–31 August 2018; pp. 1057–1064.
19. Serrano, W. Fintech Model: The Random Neural Network with Genetic Algorithm. *Procedia Comput. Sci.* **2018**, *126*, 537–546. [CrossRef]
20. Jung, D.; Dorner, V.; Weinhardt, C.; Pusmaz, H. Designing a Robo-Advisor for Risk-Averse, Low-Budget Consumers. *Electron. Mark.* **2018**, *28*, 367–380. [CrossRef]
21. Abdullah, E.M.E.; Rahman, A.A.; Rahim, R.A. Adoption of Financial Technology (Fintech) in Mutual Fund/Unit Trust Investment Among Malaysians: Unified Theory of Acceptance and Use of Technology (UTAUT). *Int. J. Eng. Technol.* **2018**, *7*, 110–118. [CrossRef]
22. Adams, J.; Hagras, H. A Type-2 Fuzzy Logic Approach to Explainable AI for Regulatory Compliance, Fair Customer Outcomes, and Market Stability in the Global Financial Sector. In Proceedings of the 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
23. Qian, H.; Wang, B.; Yuan, M.; Gao, S.; Song, Y. Financial Distress Prediction Using a Corrected Feature Selection Measure and Gradient Boosted Decision Tree. *Expert Syst. Appl.* **2022**, *190*, 116202. [CrossRef]
24. Liu, J.; Wan, Y.; Qu, S.; Qing, R.; Sriboonchitta, S. Dynamic Correlation Between the Chinese and the US Financial Markets: From Global Financial Crisis to COVID-19 Pandemic. *Axioms* **2023**, *12*, 14. [CrossRef]
25. Sun, C.-C. A Fuzzy Decision-Making Soft Model for Family Financial Planning in the Post-COVID-19 World. *Axioms* **2022**, *11*, 452. [CrossRef]
26. Pulawska, K. Financial Stability of European Insurance Companies During the COVID-19 Pandemic.*J. Risk Financ. Manag.* **2021**, *14*, 266. [CrossRef]
27. Perfilieva, I.; Yarushkina, N.; Afanasieva, T.; Romanov, A. Time Series Analysis Using Soft Computing Methods. *Int. J. Gen. Syst.* **2013**, *42*, 687–705. [CrossRef]
28. Minfin. Available online: https://minfin.gov.ru/ru/document?id_4=10352 (accessed on 10 October 2024).
29. SCF Group. Available online: https://www.scf-group.com/en/investors/disclosure/rasfinancialstatements/?year=2021 (accessed on 10 October 2024).
30. Minfin. Available online: https://minfin.gov.ru/en/policy_issues/account?id_4=119278-federal_law_of_december_6_2011__402-fz_on_accounting (accessed on 10 October 2024).
31. RSHB. Available online: https://www.rshb.ru/en/investor-relations/annual-reports (accessed on 10 October 2024).
32. Credinform. Available online: https://credinform.ru/en-GB/publications/b6b324673cc2 (accessed on 10 October 2024).
33. W3C. OWL 2 Web Ontology Language Document Overview (Second Edition). Available online: https://www.w3.org/TR/owl2-overview/ (accessed on 10 October 2024).
34. W3C. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Available online: https://www.w3.org/submissions/SWRL/ (accessed on 10 October 2024).
35. W3C. Pellet: Open Source Java Base OWL 2 Reasoner. Available online: https://www.w3.org/2001/sw/wiki/Pellet (accessed on 10 October 2024).
36. Filippov, A.A.; Romanov, A.A. Construction of a Fuzzy Logical Inference System Based on Semantic Web Technologies. In Proceedings of the 2024 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 8–14 September 2024.
37. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.F. *The Description Logic Handbook: Theory, Implementation, and Applications*; Cambridge University Press: New York, NY, USA, 2003.
38. Rudolph, S. Foundations of Description Logics. In *Reasoning Web International Summer School*; Springer: Berlin/Heidelberg, Germany, 2011.
39. Zadeh, L.A. Fuzzy Logic. *Computer* **1988**, *21*, 83–93. [CrossRef]

40. Zadeh, L.A. The Concept of a Linguistic Variable and Its Application to Approximate Reasoning—I. *Inf. Sci.* **1975**, *8*, 199–249. [CrossRef]
41. Hassanpour, S.; O'Connor, M.J.; Das, A.K. Visualizing Logical Dependencies in SWRL Rule Bases. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 259–272.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

mathematics

MDPI

Article

A Weights Direct Determination Neural Network for International Standard Classification of Occupations

Dimitris Lagios [1], Spyridon D. Mourtas [2,3,*], Panagiotis Zervas [1] and Giannis Tzimas [1]

[1] Data and Media Laboratory, Department of Electrical and Computer Engineering, University of Peloponnese, 26334 Patras, Greece
[2] Department of Economics, Mathematics-Informatics and Statistics-Econometrics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece
[3] Laboratory "Hybrid Methods of Modelling and Optimization in Complex Systems", Siberian Federal University, Prosp. Svobodny 79, 660041 Krasnoyarsk, Russia
* Correspondence: spirmour@econ.uoa.gr

Abstract: Multiclass classification is one of the most popular machine learning tasks. The main focus of this paper is to classify occupations according to the International Standard Classification of Occupations (ISCO) using a weights and structure determination (WASD)-based neural network. In general, WASD-trained neural networks are known to overcome the drawbacks of conventional back-propagation trained neural networks, such as slow training speed and local minimum. However, WASD-based neural networks have not yet been applied to address the challenges of multiclass classification. As a result, a novel WASD for multiclass classification (WASDMC)-based neural network is introduced in this paper. When applied to two publicly accessible ISCO datasets, the WASDMC-based neural network displayed superior performance across all measures, compared to some of the best-performing classification models that the MATLAB classification learner app has to offer.

Keywords: neural networks; weights and structure determination; multiclass classification; international standard classification of occupations; machine learning

MSC: 68T10; 91B40; 65F20

1. Introduction

In machine learning (ML), the classification problem arises in a variety of fields, including economics and finance [1], medicine [2], medical [3], and engineering [4]. Multiclass classification is a significant issue in these fields. Each instance in the learning set is a member of a different set of labels that were previously established for multiclass classification. The goal of supervised classification techniques is to build a learning model from a training set of labeled data so that it can classify new objects with unknown labels [5]. The multiclass classification problem can be solved by extending the binary classification problem using several ML approaches, such as NNs, decision trees, k-nearest neighbor, naive Bayes, and support vector machines [6]. The main focus of this paper is to handle multiclass classification tasks using NNs.

On the one hand, NNs are frequently employed for classification and regression problems, and they have been effectively used in a variety of fields, including but not limited to, economics and finance, medicine, and engineering. Particularly, in the field of economics and finance, NNs have been used to optimize portfolios [7], analyze time series [8], stabilize stochastic exchange rate dynamics [9], and forecast a variety of macroeconomic factors [10]. In the field of medicine, NNs have been used to diagnose breast cancer [11], lung cancer [12], flat foot [13], and to classify diabetic retinopathy [14], whereas in the field of engineering, they have been used to stabilize feedback control systems [15], track mobile

objects [16], analyze the performance of solar systems [17], and forecast the flow behavior of alloy [18].

On the other hand, social science research usually involves multiclass classification tasks, such as characterizing occupational mobility [19], conducting case-control studies in healthcare [20], examining the relationship between cancer and changes in occupational characteristics [21], and evaluating jobs' potential for telework [22]. Particularly, a multitude of classification systems are the foundation for the systematic surveillance of any population, from the general populace of the world to the people who make up a small- or medium-sized business or community [23]. The occupation (job title) or the industry a person works in can be used to reflect occupational exposure for a variety of work-related topics, such as administrative usage, employment statistics, social sciences, international trade and commerce, comprising surveillance and analytical methodologies [24]. There are numerous national and international classification schemes for these concepts, which are periodically updated [25]. International Standard Classification of Occupations (ISCO), in particular, was created to make it easier to compare occupational statistics across borders and to act as a guide for nations creating or updating their own national occupational classification systems. It has been developed to serve broad administrative and research purposes, whereas the international community fully endorses it as a recognized benchmark for international labor statistics [26]. It is worth mentioning that the most recent version of ISCO, known as ISCO-08, was adopted in 2008. Some case studies on occupational classification can be found in the Finnish Register of Occupational Diseases [27] and on the names of music professions [28].

The task of classifying occupations using NNs has become increasingly common during the past few years [29–33]. In particular, a mathematical model that uses transformer NNs and ISCO-08 to connect education and occupations is proposed in [29]. By using NNs and ISCO-08, ref. [30] links the necessary skills that have surfaced in the labor market from a demand viewpoint and occupations. Ref. [31] proposes a method for analyzing job titles and embedding them, and then utilizes NN models to demonstrate that they outperform humans and establish the baseline accuracy for identifying occupations based on ISCO-08. Using statistical ML and ISCO-08, a comprehensive occupational and economic framework for wage prediction is constructed in [32]. Ref. [33] describes a method to assess and choose embeddings from a large text corpus while maintaining the co-hyponyms links synthesized from ISCO-08 taxonomy, and uses the chosen embeddings as features in order to classify co-hyponym associations through NNs.

In this paper, we will use a feed-forward NN to handle multiclass classification tasks in social sciences. Back-propagation algorithms have a long history of being used to train feed-forward NNs, where the structure of the network is iteratively adjusted. However, a feature that their forerunners lacked is provided by recently introduced weights and structure determination (WASD) training algorithms. In particular, the weights direct determination (WDD) process, which is a component of all WASD algorithms, makes it easier to directly compute the ideal set of weights, preventing one from becoming stuck in local minima and ultimately helping to attain lower computational complexity [34]. We thus develop a 3-layer feed-forward WASD for multiclass classification (WASDMC)-based NN. The novel WASDMC algorithm uses the WDD process combined with a power maxout activation function to train the WASDMC-based NN. The prominent multiclass classification task of classifying occupations based on ISCO-08 is taken into consideration to examine the performance of the WASDMC-based NN. In particular, when applied to two publicly accessible ISCO datasets, the WASDMC-based NN displayed superior performance across all measures, compared to some of the best-performing classification models that the MATLAB classification learner app has to offer.

The main contribution of this paper is the use of a WASD-based NN to address multiclass classification tasks for the first time and the development of a new activation function specifically created for WASD-based NNs. The key ideas of this work can be summed up as follows:

- A novel 3-layer feed-forward WASDMC-based NN for multiclass classification tasks is presented.
- A novel power maxout activation function is employed to increase the accuracy of WASDMC-based NN.
- Two publicly accessible ISCO datasets are considered, and the WASDMC-based NN performance is compared to some of the best-performing classification models that the MATLAB classification learner app has to offer.
- A different power activation function is also used to show how the proposed power maxout activation function behaves differently from other activation functions.

The structure of the paper is described in the sections that follow. An overview of the final structure and justification for the WASDMC-based NN is given at the start of Section 2. The creation of the power maxout activation function and the design of the WDD procedure ensue. The section concludes with a detailed explanation of the WASDMC algorithm and the entire training procedure. The two ISCO datasets' data preparation for use with the WASDMC-based NN is described in Section 3. The results of the WASDMC-based NN on the two ISCO datasets are shown in Section 4, and its performance is evaluated in comparison to that of other well-known models. A brief summary and helpful information for the MATLAB package that has been made accessible on GitHub are also included in Section 4 to support the quality and computational utility of this work. Final observations are provided in Section 5.

2. The WASDMC-Based NN Model

This section describes the 3-layer feed-forward WASDMC-based NN for multiclass classification tasks. Its structure, which has m input and n hidden layer neurons, can be as shown in Figure 1. In particular, Layer 1 is the input layer, receiving the input values $X_1, X_2, \ldots, X_m$, and allocating them to the relevant neuron in Layer 2, which has a maximum number of activated neurons of n, with equal weight 1, whereas Layer 3 is the output layer and has one activated neuron. The WDD process is used to acquire the weights $W_i, i = 1, 2, \ldots, n-1$ in the neurons that connect Layer 2 and Layer 3 neurons. The NN model can achieve low hidden layer utilization by employing the WASDMC algorithm.

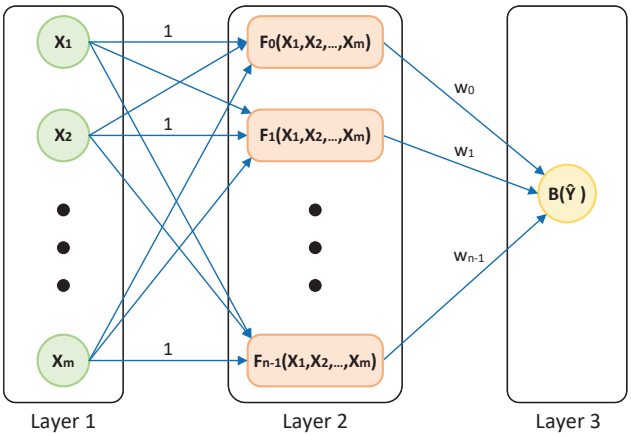


Figure 1. Structure of the WASDMC-based neural network.

2.1. The WDD Process with the Power Maxout Activation Function

The WDD process is a key component of any WASD algorithm since it eliminates the need for time-consuming, frequently unreliable iterative computations to find the ideal weights matching to the present hidden layer structure. Apparently, compared to conventional weight determination methods, the WDD process helps to achieve both speed and lower computational complexity, while avoiding some of the associated difficulties [34]. Here, detailed explanations of crucial theoretical foundations and analyses are

offered for the WASDMC-based NN design. The Taylor polynomial (TAPO) approximation theorem [35] is first established as follows.

Theorem 1. *When a target function $f(\cdot)$ has the $(K+1)$-order continuous derivative on the interval $[a_1, a_2]$ and K is a non-negative integer, the following applies:*

$$f(x) = A_K(x) + E_K(x), \quad x \in [a_1, a_2], \tag{1}$$

where $A_K(x)$ and $E_K(x)$, respectively, denote the K-order TAPO approximation of $f(x)$ and the error term.

Considering the value of the r-order derivative of $f(x)$, which is denoted by $f^{(r)}(u)$ at the point u, $f(x)$ can be approximately represented as seen below:

$$f(x) \approx A_K(x) = \sum_{r=0}^{K} \frac{f^{(r)}(u)}{r!}(x-u)^r, \quad u \in [a_1, a_2], \tag{2}$$

where $r!$ denotes the factorial of r.

Proposition 1. *To approximate multivariable functions, one may utilize the TAPO approximation Theorem 1. Consider the target function $f(x_1, x_2, \ldots, x_h)$ with h variables and $(K+1)$-order continuous partial derivatives in an origin's neighborhood $(0, \ldots, 0)$. Then, the K-order TAPO $A_K(x_1, x_2, \ldots, x_h)$ about the origin is the following:*

$$A_K(x_1, x_2, \ldots, x_h) = \sum_{r=0}^{K} \sum_{r_1+\cdots+r_h=r} \frac{x_1 \cdots x_h}{r_1 \cdots r_h} \left(\frac{\partial^{r_1+\cdots+r_h} f(0, \cdots, 0)}{\partial x_1^{r_1} \cdots \partial x_h^{r_h}} \right), \tag{3}$$

where $r_1, r_2, \ldots, r_h$ are non-negative integers.

The WDD process limits the input data to only be real numbers. As a result, we assume the input $X = [X_1, X_2, \ldots, X_m] \in \mathbb{R}^{1\times m}$ and the target vector $Y \in \mathbb{R}$. Based on the power activated multi-input NNs in [34], the relationship between the input variables $X_1, X_2, \ldots, X_m$ and the output target Y of the NN can be represented by the nonlinear function shown below:

$$f(X_1, X_2, \ldots, X_m) = Y. \tag{4}$$

Therefore, according to Proposition 1, the K-order TAPO $A_K(X_1, X_2, \ldots, X_m)$ can map (4) as follows:

$$A_K(X_1, X_2, \ldots, X_m) = \sum_{r=0}^{n-1} z_r w_r, \tag{5}$$

where $z_r = R_r(X_1, X_2, \ldots, X_m) \in \mathbb{R}^{1\times mn}$ is a power activation function, $w_r \in \mathbb{R}^{mn}$ is the weight that refers to z_r, and r indicates both the power value and the number of the hidden layer neurons.

On WASD-based NNs, a variety of activation functions are used, including Chebyshev and Euler polynomials, signum, power, sine, and square wave [36–40]. However, because each activation function has its own combination of empirical and mathematical characteristics, it might be more appropriate for a certain problem. When dealing with multiclass classification tasks, the maxout activation function [41] is regarded as one of the most efficient activation functions. After modifying the maxout activation function to be employed in (5), the following elementwise power maxout activation function is proposed:

$$R_r(X_i) = \begin{cases} X_i^r & , X_i^r = \max(X^{\odot r}) \\ 0 & , \text{otherwise} \end{cases}, \quad \text{for } i = 1, 2, \ldots, m, \tag{6}$$

where the superscript $()^{\odot}$ denotes the Hadamard (or elementwise) exponential.

For a given number of samples $s \in \mathbb{N}$, we assume the standardized input matrix $X = [X_1, X_2, \ldots, X_m] \in \mathbb{R}^{s\times m}$ and the target vector $Y \in \mathbb{R}^s$. Further, setting $z_{s,r} = R_r(X_1, X_2, \ldots, X_m) \in \mathbb{R}^{s\times mn}$, the input-activation matrix becomes

$$Z = \begin{bmatrix} z_{1,0} & z_{1,1} & \cdots & z_{1,n-1} \\ z_{2,0} & z_{2,1} & \cdots & z_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{s,0} & z_{s,1} & \cdots & z_{s,n-1} \end{bmatrix} \in \mathbb{R}^{s\times mn}, \tag{7}$$

and the weight vector becomes $W = [w_0, w_1, \ldots, w_{n-1}]^{\mathrm{T}} \in \mathbb{R}^{mn}$. Note that the superscript $()^{\mathrm{T}}$ denotes transposition. Then, instead of iterative weight training in conventional NNs, the weights of the WASDMC-based NN in Figure 1 are computed using the WDD process, which is stated in the following Lemma 1.

Lemma 1. *The K-order TAPO NN's steady-state weights can be directly obtained as shown below [35]:*

$$W = Z^{\dagger}Y, \tag{8}$$

where the superscript $()^{\dagger}$ *denotes pseudoinversion.*

2.2. The WASDMC Algorithm and the NN Model

The WASDMC algorithm is liable for training the NN model. To avoid overfitting, the data X must first be normalized in the range $[0, 1]$ [34]. After that, the WDD process is used on all the training samples of the data X. Particularly, setting $r = 0$, the Z matrix is constructed according to (6) and (7), and the weights of the NN are directly obtained through (8). To calculate the NN predictions $\hat{Y}$, the following formula is used:

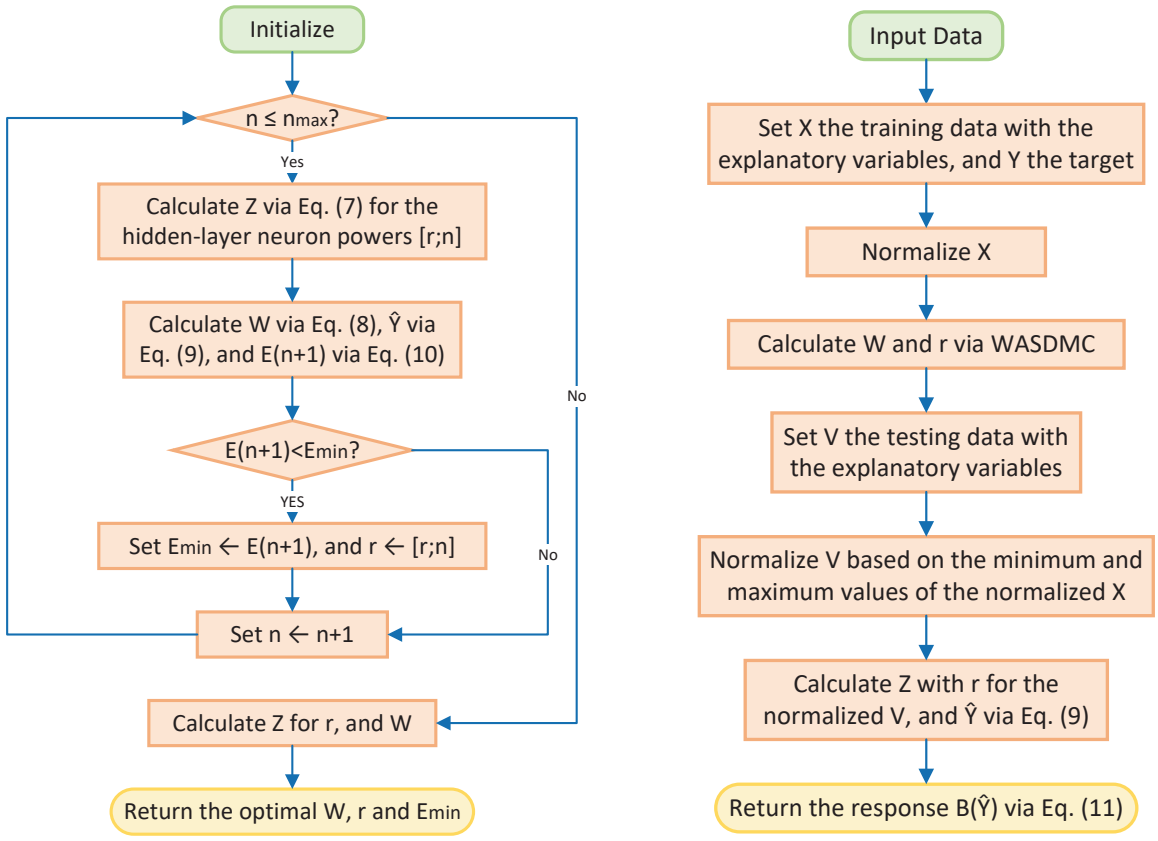
$$\hat{Y} = \mathrm{round}(ZW), \tag{9}$$

where $\mathrm{round}(\cdot)$ denotes a round function. It is significant to note that since the WDD process limits the input data to only be real numbers, the output of ZW is also a real number. Given this, the round function in (9) is used to ensure that this output rounds to the closest positive integer that corresponds to a single class. Then, the mean absolute error (MAE) between the predictions $\hat{Y}$ and the target value Y is measured. Keep in mind that the MAE is a measurement of errors representing the same phenomenon between paired observations and is typically used as a loss function in ML for classification tasks. Furthermore, MAE values that are closer to zero are preferable, and they are calculated as follows:

$$\mathrm{MAE} = \frac{1}{s}\sum_{q=1}^{s} \left|Y_q - \hat{Y}_q\right|, \tag{10}$$

where s implies the number of samples. In this way, the WASDMC algorithm validates the data in the entire training set X.

The WASDMC algorithm then iteratively selects the optimal activation function for each r depending on the MAE. Given a maximum number of hidden-layer neurons $n_{\max}$, the highest performing powers r in (6) are picked iteratively from 0 to $n_{\max} - 1$. Specifically, a new hidden-layer neuron forms under a given n if the MAE for that n is lower than the previous best MAE, whereas a n is bypassed and not included in the hidden-layer neurons if the MAE for that n is higher than the previous best MAE. In this manner, the WASDMC algorithm is able to maintain the lowest number of hidden-layer neurons in the NN while lowering the total MAE of the NN model. It is important to mention that $\max(X^{\odot r}) = 1$ because of the normalization in the range $[0, 1]$. As a consequence, the maximum number of hidden-layer neurons is limited to $n_{\max} = 2$. This happens as a result of the properties of (6), which lead to every matrix Z for $n > 2$ being the same as the matrix Z for $n = 2$. The diagram in Figure 2a shows the comprehensive process of the WASDMC algorithm.



(**a**) The WASDMC algorithm. (**b**) Process for modeling and predicting.

Figure 2. The WASDMC algorithm and the process for modeling and predicting with the WASDMC-based NN model.

Based on the aforementioned procedures, the following are the main steps for the complete process of training and predicting the WASDMC-based NN model of Figure 1. First, the NN model receives the standardized input X of s samples in Layer 1. Then, the WASDMC algorithm discovers the optimal number of hidden-layer neurons n^* as well as the optimal power of the activation function at each hidden-layer neuron in Layer 2. Last, $\hat{Y}$ is calculated through 9, and the following elementwise function is employed in Layer 3:

$$B(\hat{Y}_i) = \begin{cases} \max(Y) & , \hat{Y}_i > \max(Y) \\ \hat{Y}_i & , \min(Y) \leq \hat{Y}_i \leq \max(Y) \\ \min(Y) & , \hat{Y}_i < \min(Y) \end{cases}, \quad \text{for } i = 1, 2, \ldots, s, \tag{11}$$

The diagram in Figure 2b shows the comprehensive process for modeling and predicting with the WASDMC-based NN model. It is important to mention that the normalization of the input data is described in the following Section 3. Additionally, since the result of (9) directly pertains to a single class, denormalization is not necessary.

3. Data Preparation

This section describes the two ISCO datasets' data preparation for use with the WASDMC-based NN. It is worth mentioning that ISCO is often built on the concepts of skill and job, i.e., the ability to do the activities and duties of a certain job, which may require some amount of formal training. A job is defined as a specific type of work performed by one person and serves as the statistical unit of ISCO-08. In other words, ISCO is an instrument for dividing jobs into a set of groups that are clearly defined in accordance with the tasks and responsibilities performed in the job, and it is designed to be used in a range

of client-oriented applications as well as statistical applications. From a social, economic, and medical standpoint, this division is beneficial. Additionally, jobs can be classified into 436 4-digit unit groups, 130 3-digit minor groups, 43 2-digit sub-major groups, and ten major 1-digit groups using the ISCO four-level hierarchically organized classification system. In our approach, the classification tasks solely took into account the 4-digit unit groups.

For the performance assessment of the WASDMC-based NN, two ISCO datasets are employed. The first dataset is taken from the International Labor Organization (ILO) and can be accessed at [42]. We shall refer to this dataset as ILO-ISCO for simplicity. The second dataset is taken from [43] and includes data extracted from online sources (OS), pre-processed, and hand-annotated following the ISCO taxonomy. We shall refer to this dataset as OS-ISCO for simplicity. On the one hand, there are 436 different classes of occupations in ILO-ISCO, and each class has a variety of job titles that fit within it. On the other hand, there are 2581 different job titles in OS-ISCO, and each one is associated with one of 149 different classes of occupations.

To train and test the NN model, the input data can only be real numbers due to restrictions imposed by the WDD process. However, both of the datasets in our study contain strings; therefore, the data must be adequately processed and prepared before being input into the WASDMC-based NN model. The process that each dataset should go through is described in the following steps $\mathbf{S_1}$ to $\mathbf{S_5}$.

$\mathbf{S_1}$: It is necessary to develop a primary vocabulary for each class. To accomplish this, the different job titles that all fall under one class must be grouped together, and then they must be tokenized according to rules based on Unicode Standard Annex #29 [44], with all letters converted to lowercase and punctuation removed. As a result, given that there are m classes, we create the structure vector $V = [v_1, v_2, \ldots, v_m]$, where $v_j, j = 1, \ldots, m$ is the vocabulary of class j.

$\mathbf{S_2}$: Consider that the NN model will be tested using s in number samples, each of which contains one job title. It is necessary to develop a vocabulary for each job title. To accomplish this, each job title must be tokenized according to rules based on Unicode Standard Annex #29 [44], with all letters converted to lowercase and punctuation removed. As a result, we create the structure vector $G = [g_1, g_2, \ldots, g_s]$, where $g_i, i = 1, \ldots, s$ is the vocabulary of some class.

$\mathbf{S_3}$: Considering that some words of a vocabulary in G may belong to several different class vocabularies in V, we create an input matrix X for training and testing the model that includes the percentages of similarity between the vocabularies in G and the vocabularies in V. Particularly, the following process is taken into consideration to create the input matrix $X \in \mathbb{R}^{s \times m}$. Assume the aforementioned structure vectors V and G, where the vocabularies $v_j, j = 1, \ldots, m$ and $g_i, i = 1, \ldots, s$ have k_j and h_i amount of words, respectively, and that strcmp$(\cdot)$ is a function that returns 1 (true) if the input strings are identical and 0 (false) otherwise (see [45]). Then, the percentage of the vocabulary v_j that matches the vocabulary g_i is

$$P_v(i,j) = \frac{1}{k_j} \sum_{q=1}^{k_j} \text{strcmp}(g_i, v_j(q)), \tag{12}$$

where $v_j(q)$ refers to the qth element (word) of the vocabulary v_j, and the percentage of the vocabulary g_i that matches the vocabulary v_j is

$$P_g(i,j) = \frac{1}{h_i} \sum_{q=1}^{k_j} \text{strcmp}(g_i, v_j(q)). \tag{13}$$

This led us to determine that the ij element of X should represent the average of (12) and (13) as follows:

$$X(i,j) = \frac{1}{2}(P_v(i,j) + P_g(i,j)), \tag{14}$$

which takes values in the range $[0, 1]$.

S_4: To train the NN model, we create the input matrix X_{tr}. Given that there are m distinct classes, setting $G = V$ in step **S_3** will create an input matrix $X_{tr} \in \mathbb{R}^{m \times m}$. As a result, X_{tr} will include the percentages of similarity between the vocabularies in V. Additionally, we create the target vector $Y_{tr} = [1, 2, \ldots, m]^{\mathrm{T}} \in \mathbb{R}^m$, where each number from 1 to m corresponds to a single class.

S_5: To test the NN model, we create the input matrix X_{te}. Given that there are s in number samples, step **S_3** will create an input matrix $X_{te} \in \mathbb{R}^{s \times m}$. As a result, X_{te} will include the percentages of similarity between the vocabularies in G and the vocabularies in V. Additionally, the numbers that the classes have taken in Y_{tr} should be used to create the target vector $Y_{te} \in \mathbb{R}^s$.

It should be emphasized that steps **S_1** to **S_5** are a heuristic approach that accomplishes the problem's goals, namely converting texts to numbers and standardizing the input matrix. By employing the steps **S_1** to **S_5**, the following input matrices and target vectors will be created for training and testing the models for each dataset. In the case of ILO-ISCO, we will create an input matrix $X_{tr} \in \mathbb{R}^{436 \times 436}$ and a target vector $Y_{tr} \in \mathbb{R}^{436}$ since there are 436 different classes of occupations. Additionally, we will use the first and the last job titles of each class as test data. As a result, we will create an input matrix $X_{te} \in \mathbb{R}^{872 \times 436}$ and a target vector $Y_{te} \in \mathbb{R}^{872}$. In this way, there are 1308 samples in total, of which 33.3% are used in the training set and 66.7% in the testing set. In the case of OS-ISCO, we will create an input matrix $X_{tr} \in \mathbb{R}^{149 \times 149}$ and a target vector $Y_{tr} \in \mathbb{R}^{149}$ since there are 149 different classes of occupations. Additionally, we will use all the job titles of each class as test data. As a result, we will create an input matrix $X_{te} \in \mathbb{R}^{2581 \times 149}$ and a target vector $Y_{te} \in \mathbb{R}^{2581}$ since there are 2581 different job titles. In this way, there are 2730 samples in total, of which 5.5% are used in the training set and 94.5% in the testing set. It is important to mention that the number of samples in the training set must always equal the number of different classes of occupations, whereas the number of samples in the testing set can be chosen at the user's discretion. In light of this, the user may decide on the training-to-testing dataset ratio.

4. NNs Performance Assessment

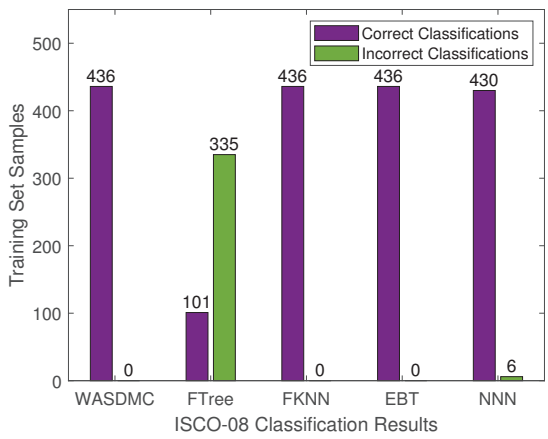
In this section, the performance of the WASDMC-based NN is examined using the multiclass classification task of classifying occupations based on ISCO-08. Using the ILO-ISCO and OS-ISCO datasets described in Section 3, the WASDMC-based NN performance is compared to some of the best-performing classification models that the MATLAB classification learner app has to offer. These classification models are fine tree (FTree), fine K-nearest neighbors (FKNNs), the ensemble bagged trees (EBTs), and the narrow neural network (NNN). More information on the algorithms that these classification models employ and how they are implemented can be found in [46]. Additionally, to demonstrate how the proposed activation function performs differently from existing activation functions, the power activation function (PAF) from [34] is also employed in place of the power maxout (6). The following elementwise function is the PAF:

$$R_r(X_i) = X_i^r, \quad \text{for } i = 1, 2, \ldots, m. \tag{15}$$

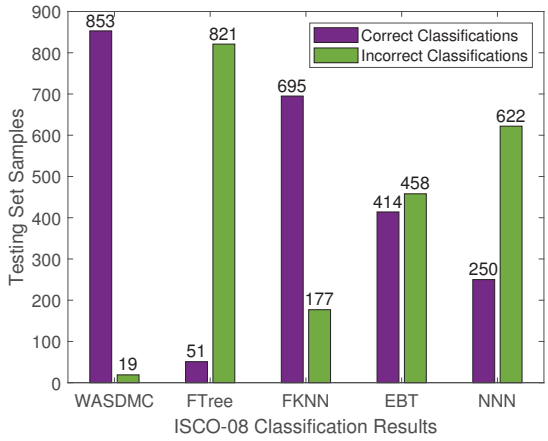
In this section, the WASDMC-PAF is the abbreviation for the WASDMC-based NN with the PAF, while the WASDMC is the abbreviation for the WASDMC-based NN with the power maxout activation function. Additionally, the WASDMC-PAF maximum number of hidden-layer neurons is set to $n_{\max} = 20$, while the FTree, FKNN, EBT, and NNN models' training parameters are kept at their default configuration in the MATLAB classification learner app. It is worth mentioning that you can download the full development and implementation of the computational methods described in this paper from GitHub at [47]. The concepts and algorithms described in Sections 2 and 3 were used in this link to construct a MATLAB package that can handle the multiclass classification tasks required by this

section. Additionally, the MATLAB package offers extensive installation guidelines and thorough implementation.

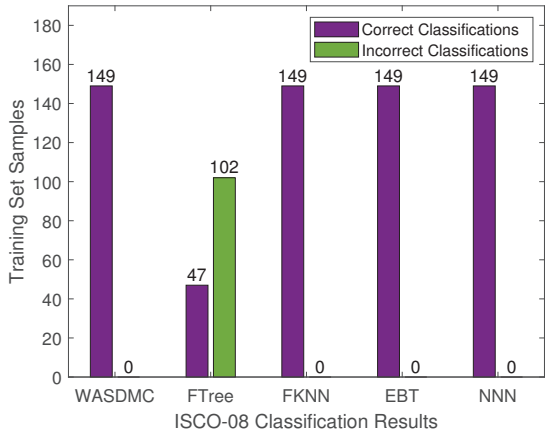
In the case of ILO-ISCO, the training set's input matrix $X_{tr} \in \mathbb{R}^{436\times436}$ and the target vector $Y_{tr} \in \mathbb{R}^{436}$ from Section 3 were used for training the classification models. Their results on the training set are presented in Figure 3a, where we observe that the WASDMC, FKNN and EBT all have perfect classification scores. That is, these models classified correctly all the 436 samples of the training set. The WASDMC-PAF performs the worst and the FTree performs the second-worst, while both models feature more incorrect classifications than correct ones. Furthermore, the testing set's input matrix $X_{te} \in \mathbb{R}^{872\times436}$ and the target vector $Y_{te} \in \mathbb{R}^{872}$ from Section 3 were used for testing the classification models. Their results on the 872 samples of the testing set are presented in Figure 3b, where we observe that the WASDMC has the best performance, with only 19 incorrect classifications, and FKNN has the second-best performance, with 177 incorrect classifications. The WASDMC-PAF performs the worst, with 852 incorrect classifications, the FTree performs the second worst, with 821 incorrect classifications, the NNN performs the third worst, with 622 incorrect classifications, and the EBT performs the fourth worst, with 458 incorrect classifications. Notice that the WASDMC-PAF, FTree, NNN and EBT have more incorrect classifications than correct ones.



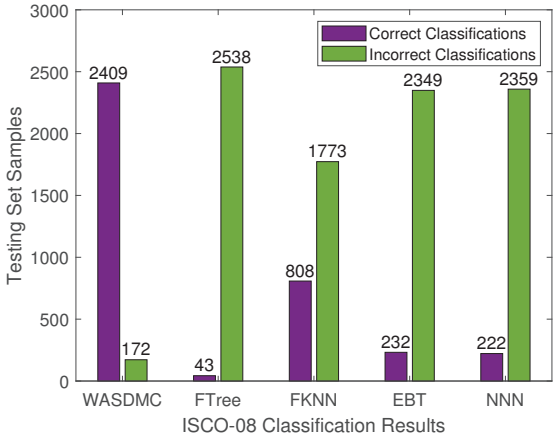
(**a**) Classification of ILO-ISCO's training set.



(**b**) Classification of ILO-ISCO's testing set.



(**c**) Classification of OS-ISCO's training set.



(**d**) Classification of OS-ISCO's testing set.

Figure 3. The classification of the ILO-ISCO and OS-ISCO training and testing sets by NNs.

In the case of OS-ISCO, the training set's input matrix $X_{tr} \in \mathbb{R}^{149\times149}$ and the target vector $Y_{tr} \in \mathbb{R}^{149}$ from Section 3 were used for training the classification models. Their results on the training set are presented in Figure 3c, where we observe that the WASDMC, FKNN, EBT and NNN all have perfect classification scores. That is, these models classified

correctly all the 149 samples of the training set. The WASDMC-PAF performs the worst and the FTree performs the second worst, while both models feature more incorrect classifications than correct ones. Furthermore, the testing set's input matrix $X_{te} \in \mathbb{R}^{2581\times149}$ and the target vector $Y_{te} \in \mathbb{R}^{2581}$ from Section 3 were used for testing the classification models. Their results on the 2581 samples of the testing set are presented in Figure 3d, where we observe that the WASDMC has the best performance, with only 172 incorrect classifications, while the rest of the models have more incorrect classifications than correct ones. Particularly, the FKNN has the second-best performance, with 1773 incorrect classifications, the EBT has the third-best performance, with 2349 incorrect classifications, the NNN has the fourth-best performance, with 2359 incorrect classifications, the FTree has the second-worst performance, with 2538 incorrect classifications, and the WASDMC-PAF has the worst performance, with 2566 incorrect classifications.

To statistically assess the classification models' performance, their outputs on the testing sets of ILO-ISCO and OS-ISCO are subjected to a number of performance metrics in Tables 1 and 2, respectively. These performance metrics are the MAE, accuracy, sensitivity, specificity, precision, false positive rate (FPR), F-score, Matthews correlation coefficient (MCC), and Cohen's κ. In particular, the accuracy is the proportion of samples that were correctly classified to the total number of samples, while the specificity is the proportion of correctly classified negative samples, and the sensitivity (or recall) is the proportion of correctly classified positive samples. As a result, sensitivity and specificity can be thought of as two different types of accuracy, with the first being relevant to actual positive samples and the second to actual negative samples. Moreover, precision shows the proportion of correctly classified positive samples to all positively predicted samples, sensitivity shows the correlation between observed and predicted classifications, MCC shows the proportion of correctly classified negative samples, FPR shows the proportion of incorrectly classified negative samples, and Cohen's κ is used to measure inter-rater reliability for categorical items. More information and greater analysis about these metrics are provided in [48,49]. Last, the training accuracy (TA) as well as the average time consumption (TC) and the memory usage (MU) for training the NN models are also included in these tables.

Table 1. The NN models' statistics on the testing set of ILO-ISCO.

	Neural Network Models					
ILO-ISCO	**WASDMC**	**WASDMC-PAF**	**FTree**	**FKNN**	**EBT**	**NNN**
TC	0.1882 s	1.0407 s	4.0399 sec	1.0738 s	14.8563 s	42.8813 s
MU	4.6925 MB	4.6925 MB	7.5769 MB	6.2226 MB	92.3990 MB	6.3485 MB
TA	1	0.0229	0.2317	1	1	0.9794
MAE	1.2580	86.1238	162.0447	10.4610	31.7133	81.8543
Accuracy	0.9782	0.0229	0.0584	0.7970	0.4782	0.2637
Sensitivity	0.9858	0.4417	0.9160	0.9285	0.7833	0.6370
Specificity	0.9999	0.9977	0.9978	0.9995	0.9988	0.9983
Precision	0.9782	0.0229	0.0584	0.7970	0.4782	0.2637
FPR	5×10^{-5}	0.0022	0.0021	4×10^{-4}	0.0011	0.0016
F-score	0.9756	0.5678	0.0727	0.7892	0.4900	0.4103
MCC	0.9821	0.4462	0.9165	0.8942	0.7719	0.6568
Cohen's κ	0.9782	0.0207	0.0563	0.7966	0.4770	0.2621

The NN models' statistics on the testing set of ILO-ISCO are presented in Table 1. Therein, we observe that the WASDMC performs better than the WASDMC-PAF, FTree, FKNN, EBT and NNN models. Particularly, the WASDMC is about 6 times faster than the WASDMC-PAF, the second-fastest model, and about 230 times faster than the NNN, the slowest model. The MUs of WASDMC and WASDMC-PAF are identical, and they are each about 1.3 times lower than the MU of FKNN, which is the third-lowest MU, and about 20 times lower than the MU of EBT, which is the highest MU. The TAs of WASDMC, FKNN and EBT are identical, and they are each about 43 times higher than the TA of WASDMC-PAF, which is the lowest TA. The MAE of WASDMC is about 8 times lower than the MAE of FKNN, which is the second-lowest MAE, and about 130 times lower

than the MAE of FTree, which is the highest MAE. The accuracy of WASDMC is about 1.2 times higher than the accuracy of FKNN, which is the second-highest accuracy, and about 42 times higher than the accuracy of WASDMC-PAF, which is the lowest accuracy. Furthermore, the WASDMC has the highest sensitivity, specificity, precision, F-score and Cohen's κ values, while the FKNN has the second-highest values in these metrics. The WASDMC-PAF has the lowest sensitivity, specificity, precision and Cohen's κ values, and the FTree has the lowest F-score value. The FPR of WASDMC is about 8 times lower than the FPR of FKNN, which is the second-lowest FPR, and about 44 times lower than the FPR of WASDMC-PAF, which is the highest FPR. Last, the WASDMC has the highest MCC, while the FKNN has the second-highest, and the WASDMC-PAF has the lowest.

Table 2. The NN models' statistics on the testing set of OS-ISCO.

	Neural Network Models					
OS-ISCO	**WASDMC**	**WASDMC-PAF**	**FTree**	**FKNN**	**EBT**	**NNN**
TC	0.0138 s	0.0846 s	0.5327 s	0.2275 s	2.6160 s	2.040 s
MU	3.3539 MB	3.3539 MB	3.7661 MB	3.5534 MB	14.8418 MB	3.6103 MB
TA	1	0.0403	0.3154	1	1	0.9933
MAE	2.0987	68.7791	56.5947	19.5877	36.8442	38.7617
Accuracy	0.9333	0.0058	0.0166	0.3130	0.0840	0.0867
Sensitivity	0.9170	0.6779	0.7612	0.5792	0.3425	0.2863
Specificity	0.9995	0.9933	0.9934	0.9953	0.9938	0.9938
Precision	0.9633	0.0402	0.1556	0.7699	0.4661	0.3967
FPR	4×10^{-4}	0.0067	0.0066	0.0046	0.0061	0.0061
F-score	0.9222	0.3490	0.1143	0.5045	0.3281	0.4134
MCC	0.9293	0.6798	0.7796	0.5751	0.3347	0.3095
Cohen's κ	0.9278	0.0023	0.0145	0.3094	0.0818	0.0810

The NN models' statistics on the testing set of OS-ISCO are presented in Table 2. Therein, we observe that the WASDMC performs better than the WASDMC-PAF, FTree, FKNN, EBT and NNN models. In particular, the WASDMC is about 6 times faster than the WASDMC-PAF, the second-fastest model, and about 190 times faster than the EBT, the slowest model. The MUs of WASDMC and WASDMC-PAF are identical, and they are each slightly lower than the MU of FKNN, which is the third-lowest MU, and about 4.4 times lower than the MU of EBT, which is the highest MU. The TAs of WASDMC, FKNN and EBT are identical, and they are each about 25 times higher than the TA of WASDMC-PAF, which is the lowest TA. The MAE of WASDMC is about 9 times lower than the MAE of FKNN, which is the second-lowest MAE, and about 32 times lower than the MAE of WASDMC-PAF, which is the highest MAE. The accuracy of WASDMC is about 3 times higher than the accuracy of FKNN, which is the second-highest accuracy, and about 160 times higher than the accuracy of WASDMC-PAF, which is the lowest accuracy. Furthermore, the WASDMC has the highest specificity, precision, F-score and Cohen's κ values, while the FKNN has the second-highest values in these metrics. The WASDMC-PAF has the lowest specificity, precision and Cohen's κ values, and the FTree has the lowest F-score value. The WASDMC has the highest sensitivity, while the FTree has the second-highest sensitivity, and the NNN has the lowest sensitivity. The FPR of WASDMC is about 11 times lower than the FPR of FKNN, which is the second-lowest FPR, and about 16 times lower than the FPR of WASDMC-PAF, which is the highest FPR. Last, the WASDMC has the highest MCC, while the FKNN has the second highest, and the WASDMC-PAF has the lowest.

In general, the WASDMC model functioned brilliantly in handling multiclass classification tasks, while its performance in contrast to WASDMC-PAF, FTree, FKNN, EBT and NNN models is superior. This is in keeping with the information shown in Tabs. in Tables 1 and 2. Furthermore, FKNN, which employs the k-nearest neighbors algorithm, has the second best overall performance, whereas the NNN, a NN classifier with one fully connected layer, has the second-worst overall performance. When FTree and EBT are compared, EBT performs marginally better than FTree because it integrates multiple decision trees rather than just one to generate higher predictive performance. It is important to note that despite having the weakest performance, the WASDMC-PAF had a higher F-score than FTree in both datasets and a lower MAE than FTree in the ILO-ISCO dataset. However, the F-scores do not account for true negatives, giving precision and FPR equal weight. As a consequence, measures such as the MCC and Cohen's κ are preferred to evaluate performance [48,49]. Therefore, it is evident that the FTree outperforms the WASDMC-PAF based on MCC and Cohen's κ values. When WASDMC and WASDMC-PAF are compared, their performance results differ significantly. It is important to acknowledge that the proposed power maxout activation function in (6) is superior to the PAF in (15). Aside from sharing the same MU and having the lowest TCs due to using the same WASD algorithm, the WASDMC had the best results across all measures, whereas the WASDMC-PAF had the worst results when compared to the FTree, FKNN, EBT, and NNN models. It is worth mentioning that the WDD process imposes a constraint on the WASDMC algorithm that the input data must only be real numbers. Additionally, the WASDMC algorithm's disadvantage is that it is only suitable for multiclass classification tasks. However, the WASDMC algorithm's advantages include low computational complexity, minimal use of hidden-layer neurons, low training time consumption, and high accuracy.

5. Conclusions

This paper presented the WASDMC-based NN model, which is trained by a novel WASDMC algorithm and can handle multiclass classification tasks. Applications on two publicly accessible ISCO datasets revealed that the WASDMC model handles multiclass classification tasks brilliantly, outperforming some of the best classification models available in the MATLAB classification learner app. Particularly, in both datasets, the WASDMC model had the lowest MU and was about 6 times faster than the second-fastest model and more than 190 times faster than the slowest model. In the remaining statistical metrics, the WASDMC model outperformed the other models by a factor of 1.2 to 160. Additionally, when used with the WASDMC to address classification tasks, the proposed power maxout activation function demonstrated to be much superior to the PAF.

There are certain limitations on this work as well as some recommendations.

1. The availability of the data, which constrained the extent of our analysis, was a study limitation. That is, more testing could be done, more results would be obtained, and the conclusions would be strengthened even further if more ISCO classification datasets were made accessible to the public.
2. Another option to assess and enhance the effectiveness and precision of a WASD algorithm is to incorporate a cluster validity method, comparable to the one used in conventional k-means clustering. With this method, a WASD algorithm can avoid identifying patterns in seemingly random data, which will enhance the performance of the WASD-based NN models.
3. Applications in numerous scientific fields could show how the WASDMC-based NN model is more reliable than conventional NN models.

Author Contributions: All authors (D.L., S.D.M., P.Z. and G.T.) contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Ministry of Science and Higher Education of the Russian Federation (Grant No. 075-15-2022-1121).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D. A multi-input with multi-function activated weights and structure determination neuronet for classification problems and applications in firm fraud and loan approval. *Appl. Soft Comput.* **2022**, *127*, 109351. [CrossRef]
2. Raj, R.J.S.; Shobana, S.J.; Pustokhina, I.V.; Pustokhin, D.A.; Gupta, D.; Shankar, K. Optimal feature selection-based medical image classification using deep learning model in internet of medical things. *IEEE Access* **2020**, *8*, 58006–58017. [CrossRef]
3. Rahim, T.; Usman, M.A.; Shin, S.Y. A survey on contemporary computer-aided tumor, polyp, and ulcer detection methods in wireless capsule endoscopy imaging. *Comput. Med, Imaging Graph.* **2020**, *85*, 101767. [CrossRef] [PubMed]
4. Bigdeli, B.; Pahlavani, P.; Amirkolaee, H.A. An ensemble deep learning method as data fusion system for remote sensing multisensor classification. *Appl. Soft Comput.* **2021**, *110*, 107563. [CrossRef]
5. Rácz, A.; Bajusz, D.; Héberger, K. Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification. *Molecules* **2021**, *26*, 1111. [CrossRef]
6. Venkatesan, R.; Er, M.J. A novel progressive learning technique for multi-class classification. *Neurocomputing* **2016**, *207*, 310–321. [CrossRef]
7. Mourtas, S.D.; Katsikis, V.N. Exploiting the Black-Litterman framework through error-correction neural networks. *Neurocomputing* **2022**, *498*, 43–58. [CrossRef]
8. Mourtas, S.D. A weights direct determination neuronet for time-series with applications in the industrial indices of the federal reserve bank of St. Louis. *J. Forecast.* **2022**, *14*, 1512–1524. [CrossRef]
9. Mourtas, S.D.; Katsikis, V.N.; Drakonakis, E.; Kotsios, S. Stabilization of stochastic exchange rate dynamics under central bank intervention using neuronets. *Int. J. Inf. Technol. Decis.* **2022**, 1–29. [CrossRef]
10. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D. Multi-input bio-inspired weights and structure determination neuronet with applications in European Central Bank publications. *Math. Comput. Simul.* **2022**, *193*, 451–465. [CrossRef]
11. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D. A fuzzy WASD neuronet with application in breast cancer prediction. *Neural Comput. Appl.* **2021**, *34*, 3019–3031. [CrossRef]
12. Daliri, M.R. A hybrid automatic system for the diagnosis of lung cancer based on genetic algorithm and fuzzy extreme learning machines. *J. Med. Syst.* **2012**, *36*, 1001–1005. [CrossRef]
13. Chen, L.; Huang, Z.; Li, Y.; Zeng, N.; Liu, M.; Peng, A.; Jin, L. Weight and structure determination neural network aided with double pseudoinversion for diagnosis of flat foot. *IEEE Access* **2019**, *7*, 33001–33008. [CrossRef]
14. Gayathri, S.; Krishna, A.K.; Gopi, V.P.; Palanisamy, P. Automated binary and multiclass classification of diabetic retinopathy using Haralick and multiresolution features. *IEEE Access* **2020**, *8*, 57497–57504. [CrossRef]
15. Mourtas, S.; Katsikis, V.; Kasimis, C. Feedback control systems stabilization using a bio-inspired neural network. *Eai Endorsed Trans. Robot.* **2022**, *1*, 1–13. [CrossRef]
16. Huang, H.; Fu, D.; Xiao, X.; Ning, Y.; Wang, H.; Jin, L.; Liao, S. Modified Newton integration neural algorithm for dynamic complex-valued matrix pseudoinversion applied to mobile object localization. *IEEE Trans. Ind. Infor.* **2020**, *17*, 2432–2442. [CrossRef]
17. Premalatha, N.; Arasu, A.V. Prediction of solar radiation for solar systems by using ANN models with different back propagation algorithms. *J. Appl. Res. Technol.* **2016**, *14*, 206–214. [CrossRef]
18. Huang, C.; Jia, X.; Zhang, Z. A modified back propagation artificial neural network model based on genetic algorithm to predict the flow behavior of 5754 aluminum alloy. *Materials* **2018**, *11*, 855. [CrossRef]
19. Groes, F.; Kircher, P.; Manovskii, I. The U-shapes of occupational mobility. *Rev. Econ. Stud.* **2015**, *82*, 659–692. [CrossRef]
20. Khalis, M.; Charbotel, B.; Fort, E.; Chajes, V.; Charaka, H.; Rhazi, K.E. Occupation and female breast cancer: A case-control study in Morocco. *Rev. Épidémiol. Santé Publique* **2018**, *66*, S302. [CrossRef]
21. Heinesen, E.; Imai, S.; Maruyama, S. Employment, job skills and occupational mobility of cancer survivors. *J. Health Econ.* **2018**, *58*, 151–175. [CrossRef] [PubMed]
22. Generalao, I.N. Measuring the telework potential of jobs: Evidence from the international standard classification of occupations. *Philipp. Rev. Econ.* **2021**, *58*, 92–127. [CrossRef]
23. Choi, S.B.; Yoon, J.H.; Lee, W. The modified international standard classification of occupations defined by the clustering of occupational characteristics in the Korean working conditions survey. *Ind. Health* **2020**, *58*, 132–141. [CrossRef] [PubMed]
24. Züll, C. The coding of occupations. In *GESIS Survey Guidelines*; GESIS–Leibniz Institute for the Social Sciences: Mannheim, Germany, 2016. [CrossRef]
25. Marc, D.T.; Dua, P.; Fenton, S.H.; Lalani, K.; Butler-Henderson, K. *The Health Information Workforce*; Health Informatics; Chapter Occupational Classifications in the Health Information Disciplines; Springer: Cham, Switzerland, 2021; pp. 71–78. [CrossRef]

26. Uter, W. *Kanerva's Occupational Dermatology*; Chapter Classification of Occupations; Springer: Berlin/Heidelberg, Germany, 2012. [CrossRef]
27. Aalto-Korte, K.; Koskela, K.; Pesonen, M. 12-year data on skin diseases in the Finnish register of occupational diseases II: Risk occupations with special reference to allergic contact dermatitis. *Contact Dermat.* **2020**, *82*, 343–349. [CrossRef]
28. Trzaskawka, P. Names of music professions–a linguistic case study. *Analele Univ. Ovidius Constanţa. Ser. Filol.* **2020**, *31*, 90–105.
29. Kuodytė, V.; Petkevičius, L. Education-to-skill mapping using hierarchical classification and transformer neural network. *Appl. Sci.* **2021**, *11*, 5868. [CrossRef]
30. Lovaglio, P.G.; Cesarini, M.; Mercorio, F.; Mezzanzanica, M. Skills in demand for ICT and statistical occupations: Evidence from web-based job vacancies. *Stat. Anal. Data Min.* **2018**, *11*, 78–91. [CrossRef]
31. Liu, J.; Ng, Y.C.; Gui, Z.; Singhal, T.; Blessing, L.T.M.; Wood, K.L.; Lim, K.H. Title2Vec: A contextual job title embedding for occupational named entity recognition and other applications. *J. Big Data* **2022**, *9*, 99. [CrossRef]
32. Matbouli, Y.T.; Alghamdi, S.M. Statistical machine learning regression models for salary prediction featuring economy wide activities and occupations. *Information* **2022**, *13*, 495. [CrossRef]
33. Malandri, L.; Mercorio, F.; Mezzanzanica, M.; Nobani, N. MEET-LM: A method for embeddings evaluation for taxonomic data in the labour market. *Comput. Ind.* **2021**, *124*, 103341. [CrossRef]
34. Zhang, Y.; Chen, D.; Ye, C. *Deep Neural Networks: WASD Neuronet Models, Algorithms, and Applications*; CRC Press: Boca Raton, FL, USA, 2019.
35. Zhang, Y.; Yu, X.; Xiao, L.; Li, W.; Fan, Z.; Zhang, W. Weights and structure determination of articial neuronets. In *Self-Organization: Theories and Methods*; Nova Science: New York, NY, USA, 2013.
36. Zhang, Y.; Yin, Y.; Guo, D.; Yu, X.; Xiao, L. Cross-validation based weights and structure determination of Chebyshev-polynomial neural networks for pattern classification. *Pattern Recognit.* **2014**, *47*, 3414–3428. [CrossRef]
37. Zhang, Y.; Wang, R.; Lao, W.; Deng, J. Signum-function-activated WASD neuronet and its XOR application. *Acta Sci. Nat. Univ. Sunyatseni* **2014**, *53*, 1–7.
38. Zhang, Y.; Guo, D.; Luo, Z.; Zhai, K.; Tan, H. CP-activated WASD neuronet approach to Asian population prediction with abundant experimental verification. *Neurocomputing* **2016**, *198*, 48–57. [CrossRef]
39. Zeng, T.; Zhang, Y.; Li, Z.; Qiu, B.; Ye, C. Predictions of USA presidential parties from 2021 to 2037 using historical data through square wave-activated WASD neural network. *IEEE Access* **2020**, *8*, 56630–56640. [CrossRef]
40. Zhang, Y.; Lao, W.; Jin, L.; Chen, T.; Liu, J. Growing-type WASD for power-activation neuronet to model and forecast monthly time series. In Proceedings of the 2013 10th IEEE International Conference on Control and Automation (ICCA), Hangzhou, China, 12–14 June 2013; pp. 1312–1317. [CrossRef]
41. Goodfellow, I.J.; Warde-Farley, D.; Mirza, M.; Courville, A.; Bengio, Y. Maxout networks. *arXiv* **2013**, arXiv:1302.4389.
42. ISCO-08 Structure and Definitions. Available online: https://www.ilo.org/ilostat-files/ISCO/newdocs-08-2021/ISCO-08/ISCO-08\EN\Structure\and\definitions.xlsx (accessed on 18 December 2022).
43. Varelas, G.; Lagios, D.; Ntouroukis, S.; Zervas, P.; Parsons, K.; Tzimas, G. Employing natural language processing techniques for online job vacancies classification. In *Artificial Intelligence Applications and Innovations. AIAI 2022 IFIP WG 12.5 International Workshops. AIAI 2022*; IFIP Advances in Information and Communication Technology; Maglogiannis, I., Iliadis, L., Macintyre, J., Cortez, P., Eds.; Springer: Cham, Switzerland, 2022; Volume 652, pp. 333–344. [CrossRef]
44. Davis, M.; Iancu, L. Unicode text segmentation. *Unicode Stand. Annex.* **2018**, *29*, 65.
45. Gupta, A.K. *Numerical Methods Using MATLAB*; MATLAB Solutions Series, Berkley; Springer Press: New York, NY, USA, 2014.
46. Kim, P. *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*; Apress: Berkeley, CA, USA, 2017. [CrossRef]
47. WASD-Based Neural Network for Multiclass Classification. Available online: https://github.com/SDMourtas/WASDMC (accessed on 21 December 2022).
48. Tharwat, A. Classification assessment methods. *Appl. Comput. Inf.* **2020**, *17*, 168–192. [CrossRef]
49. McHugh, M.L. Interrater reliability: the kappa statistic. *Biochem. Med.* **2012**, *22*, 276–282. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

mathematics

MDPI

Article

Portfolio Insurance through Error-Correction Neural Networks

Vladislav N. Kovalnogov [1], Ruslan V. Fedorov [1], Dmitry A. Generalov [1], Andrey V. Chukalin [1], Vasilios N. Katsikis [2,*], Spyridon D. Mourtas [2] and Theodore E. Simos [1,3,4,5,6]

[1] Laboratory of Inter-Disciplinary Problems of Energy Production, Ulyanovsk State Technical University, 32 Severny Venetz Street, 432027 Ulyanovsk, Russia
[2] Department of Economics, Division of Mathematics-Informatics and Statistics-Econometrics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece
[3] Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 40402, Taiwan
[4] Data Recovery Key Laboratory of Sichun Province, Neijing Normal University, Neijiang 641100, China
[5] Department of Mathematics, University of Western Macedonia, 52100 Kastoria, Greece
[6] Section of Mathematics, Department of Civil Engineering, Democritus University of Thrace, 67100 Xanthi, Greece
* Correspondence: vaskatsikis@econ.uoa.gr

Abstract: Minimum-cost portfolio insurance (MCPI) is a well-known investment strategy that tries to limit the losses a portfolio may incur as stocks decrease in price without requiring the portfolio manager to sell those stocks. In this research, we define and study the time-varying MCPI problem as a time-varying linear programming problem. More precisely, using real-world datasets, three different error-correction neural networks are employed to address this financial time-varying linear programming problem in continuous-time. These neural network solvers are the zeroing neural network (ZNN), the linear-variational-inequality primal-dual neural network (LVI-PDNN), and the simplified LVI-PDNN (S-LVI-PDNN). The neural network solvers are tested using real-world data on portfolios of up to 20 stocks, and the results show that they are capable of solving the financial problem efficiently, in some cases more than five times faster than traditional methods, though their accuracy declines as the size of the portfolio increases. This demonstrates the speed and accuracy of neural network solvers, showing their superiority over traditional methods in moderate-size portfolios. To promote and contend the outcomes of this research, we created two MATLAB repositories, for the interested user, that are publicly accessible on GitHub.

Keywords: portfolio insurance; neural networks; portfolio selection; time-varying linear programming

MSC: 68T05; 90C05; 91B28

1. Introduction and Motivation

The effects of the severe financial crisis that occurred in 2008 have characterized financial markets in recent years, followed by the associated rise in equity markets and associated decline in interest rate levels globally, up until a significant market decline in 2018 and the extremely volatile market conditions brought on by the COVID-19 crisis in the first quarter of 2020. Such a scenario suggests that it will always be difficult to predict how asset prices will behave and to make the appropriate countermoves in order to later create appropriate portfolio strategies in accordance with certain risk profiles. In these types of scenarios, portfolio insurance strategies such as guarantee funds and variable annuities are frequently used to provide investors with the promised assurances. Portfolio insurance's primary goal is to limit a portfolio's downside risk while maintaining its maximum return potential. A number of portfolio insurance optimization strategies have been proposed in recent years, using a variety of computational techniques, such as Riesz space-based techniques [1,2] and metaheuristics [3–5]. More specifically, the authors considered a

linear programming portfolio insurance problem in [1,2], an integer programming portfolio insurance problem in [3], and a nonlinear programming multiperiod portfolio insurance problem under transaction costs in [4,5].

Recently, neural network (NN) techniques, such as weights- and structure-determination-based NNs [6,7], collaborative neurodynamic optimization [8], nonlinear NNs [9], and reinforcement learning [10], have been utilized to address portfolio optimization. A number of portfolio optimization strategies through NNs have been proposed in the past, such as predicting the distribution of a financial quantity [11], predicting the portfolio Sharpe ratio [12], and integrating the return prediction of traditional time series models in portfolio formation [13]. In general, there are two ways to apply NNs to portfolio construction: limiting loss [1,4,5] and seeking profit [11–13]. This work focuses on limiting portfolio loss. It defines and explores a time-varying variation of the minimum-cost portfolio insurance (MCPI) problem as a time-varying linear programming (TLP) problem. More precisely, using real-world datasets, three different error-correction NNs (ENNs) are employed to address the financial TLP problem in continuous-time (CT). These NN solvers are the zeroing NN (ZNN), the linear-variational-inequality primal-dual NN (LVI-PDNN), and the simplified LVI-PDNN (S-LVI-PDNN). When it comes to solving TLP problems, such as the time-varying MCPI (TMCPI) problem, the ZNN approach is comparable to the LVI-PDNN and S-LVI-PDNN methods because they are all based on the error-correction principle.

The ZNN framework was created by Zhang et al. in [14] for generating online solutions to time-varying (TV) problems and it is based on the Hopfield neural network. Notice that the vast majority of ZNN-based dynamical systems, as well as the LVI-PDNN and S-LVI-PDNN, are classified as recurrent NNs (RNNs), which are utilized to locate equations' zeros. On the one hand, the ZNN approach has been extensively used in a variety of time-varying topics, with the most common being problems of generalized inversion [15,16], matrix equations systems [17,18], quadratic optimization [19,20], linear equations systems [17], and various matrix functions approximation [21,22]. On the other hand, the LVI-PDNN is widely employed in robotic applications, such as the construction of mobile robots' vision-related control systems [23] or resolution of kinematic redundancy in robot manipulators [24], and it has recently been successfully used in solving financial problems such as asset portfolio selection [20]. As a result, this article's primary goal is to efficiently solve the TMCPI problem through ENN methods (i.e., ZNN, LVI-PDNN, and S-LVI-PDNN).

The primary advantages of employing an artificial NN include generalization, fault and noise tolerance, and the capacity to predict data that have not yet been seen while saving costs and time [21]. Based on our previous experience with portfolio selection problems, we apply the advantages of ENN solvers (i.e., ZNN, LVI-PDNN, and S-LVI-PDNN) to solve the TMCPI problem within a realistic framework of financial portfolio management. Furthermore, as is well known, ENN solvers can be viewed as predictive dynamics. By employing these ENN methods, the proposed models in [15,19] achieve excellent convergence performance, while the convergence speed can be changed by altering the design parameter [21]. As a consequence, by using the ZNN, LVI-PDNN, and S-LVI-PDNN methods, the TMCPI problem can be addressed with exponential convergence performance. To be able to track the evolution of the static MCPI problem over time and to offer a form of prediction, we investigate the TMCPI problem as a continuous TLP problem.

The main contributions of this work can be summed up as follows:

- The TMCPI problem is defined and explored;
- Three novel ENN models for addressing the TMCPI problem are defined;
- We use ENN solvers on real-world datasets;
- The performance of the ENN solvers versus the conventional MATLAB solver is demonstrated and contrasted in trials using four different portfolio configurations.

The following hierarchy governs the overall organization of the document's sections. The TV variation of the MCPI problem is introduced in Section 2. Section 3 presents the

ENN solvers. Experiments in Section 4 highlight the performance and efficacy of the ENN solvers for solving the TMCPI problem in four different portfolio setup cases using daily real-world data. Additionally, information about a publicly accessible MATLAB repository on GitHub is provided in Section 4. This repository implements all of the techniques and procedures outlined in Sections 3 and 4 to promote the readability and computational value of this research. The conclusions section is Section 5.

2. Minimum-Cost Portfolio Insurance

In many situations, portfolio insurance solutions are employed to give investors promised guarantees [1,25,26]. The past literature has proposed a wide range of portfolio insurance strategies, some of which have been expanded to the dynamic setting. This section introduces a continuous-time portfolio insurance approach as an innovative technique that integrates robust processes from ENNs to provide online, thus more realistic, solutions.

TV Variation of the MCPI Problem

Reducing portfolio costs is always a priority for financial engineers. The obvious way to reduce portfolio costs is to reduce insurance costs (see [3,4]). Here, we present a TV analog of the corresponding portfolio insurance static problem, which has been described and investigated in a number of papers, including [2,27].

Note that, when we say a vector $d(t) \in \mathbb{R}^n$ or a matrix $D(t) \in \mathbb{R}^{m\times n}$, we mean a vector or a matrix that change over time t, and also, $(\cdot)^{\mathrm{T}}$ denotes matrix transposition. Assume the market $X(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T \in \mathbb{R}^n$ at time $t = 1, 2, \ldots, \mu$, where $x_i(t) \in \mathbb{R}$ is the asset's i, $i = 1, 2, \ldots, n$, return, and the initial portfolio $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T \in \mathbb{R}^n$, where θ_i, $i = 1, 2, \ldots, n$, is the ith asset. Then, the payoff of a portfolio θ is given by the formula

$$X^{\mathrm{T}}(t) \cdot \theta \tag{1}$$

Additionally, assume the insurance prices $p(t) = [p_1(t), p_2(t), \ldots, p_n(t)]^T \in \mathbb{R}^n$ where $p_i(t) \in \mathbb{R}$ is the asset's i, $i = 1, 2, \ldots, n$, insurance price. In our case, we assume that the portfolio's insurance costs include a standard price plus an asset risk rate. The function of the fixed and linear insurance prices is made up as follows if the price rates connected to asset risk are denoted by δ and the fixed price by β:

$$p_i(t) = \beta + \delta \cdot \mathrm{Var}\left[\frac{Y_i(t)}{\max(Y_i(t))}\right],\ i = 1, 2, \ldots, n, \tag{2}$$

where $Y_i(t) = [x_i(t), x_i(t-1), \ldots, x_i(t-\tau+1)]^T$ and $\mathrm{Var}[Y_i(t)]$ denotes the variance of $Y_i(t)$. The number $\tau \leq t-1$, $\tau \in \mathbb{N}$, is constant and implies the time delays. Since $Y_i(t) \in \mathbb{R}^\tau$, the risk of the i asset is determined by the variance of τ normalized prices. In this way, any asset's insurance cost is based on the amount of risk it carries, where there is a chance that the predicted return will differ from the actual return. As a result, insurance prices are rising in line with inflation.

It is important to mention that the insured payoff of the portfolio θ, under the floor price $k \in \mathbb{R}$ and the price $p(t)$, is the maximum between the payoff of θ and k, i.e., $\max\{X^{\mathrm{T}}(t) \cdot \theta, k\}$. Consequently, given an initial portfolio θ and a floor price k, the TLP formulation of the TMCPI problem is as below:

$$\min_{\phi(t)} \quad p^{\mathrm{T}}(t) \cdot \phi(t) \tag{3}$$

$$\text{s.t.} \quad -X^{\mathrm{T}}(t) \cdot \phi(t) \leq \min\{-X^{\mathrm{T}}(t) \cdot \theta, -k\} \tag{4}$$

$$0 \leq \phi(t) \leq \zeta(t), \tag{5}$$

where $\phi(t) = [\phi_1(t), \phi_2(t), \ldots, \phi_n(t)]^T \in \mathbb{R}^n$ is the optimal insured portfolio, $\zeta(t) = X^{\mathrm{T}}(t) \cdot \theta \cdot \left[\frac{1}{x_1(t)}, \ldots, \frac{1}{x_n(t)}\right]^T \in \mathbb{R}^n$, and the right part of (5), i.e., $\zeta(t)$, is the maximum merit

of each stock that an investor is allowed to keep at time t, by investing in each of them all the portfolio's θ payoff.

Moreover, by interpolating $X(t)$ and $p(t)$ into continuous functions, we transform the TMCPI problem from discrete-time (DT) to CT. As a result, considering the space of all continuous real functions $C[0, \mu - \tau - 1]$ on the closed interval $[0, \mu - \tau - 1]$, we have that $X(t),\ p(t) \in C[0, \mu - \tau - 1]$, and $\phi(t)$ becomes the online solution of the TMCPI problem (3)–(5).

3. The Error-Correction Neural Network Approach

Bearing in mind its basic role in mathematical optimization, most aspects of linear programming (LP) have been thoroughly studied over the last decades. Solutions to LP problems have been commonly used in a wide range of research and industrial applications; see for example [28,29]. In the past, typically fewer than two different types of constraints have been subject to LP problems; see for example [30]. Most studies were dedicated to the so-called static LP problems, see [31], which means that methods and algorithms designed to address such a group of LP problems are inoperative when employed to TV situations. The ZNN, LVI-PDNN, and S-LVI-PDNN are three types of ENNs that are specifically designed to zeroing time-varying equations, which have been critical in solving a variety of difficult TV problems in various scientific domains [20,21]. This section describes the ENN solvers for approaching the TMCPI problem of (3)–(5).

In what follows, we denote by $\mathbf{1}$ and $\mathbf{0}$, respectively, those vectors of $\mathbb{R}^n$ containing ones and zeros; I_n for the identity $n \times n$ matrix; and $\mathbf{0}_{m\times k}$ for the zero $m \times k$ matrix; $\odot$ signifies the element-wise (or Hadamard) product and $()^{\odot}$ the element-wise exponential; and a square diagonal matrix with the components of the vector x on the main diagonal is denoted by $\bar{x}$.

3.1. TMCPI Problem via ZNN

Since its introduction by Zhang et al. in 2001 [32], the ZNN evolution has been studied and developed as a significant class of recurrent NNs. The theoretical investigation of ZNNs has shown that they are a powerful tool for solving TV problems. According to the ZNN's design, under the linear activation function, we define an error matrix $K(t)$ which can be dynamically controlled using the formula [18]:

$$\dot{K}(t) = -\gamma K(t), \tag{6}$$

where $\gamma > 0$ is the design parameter and $\dot{(\ )}$ signifies the time derivative. $K(t)$ is forced to exponentially converge to the zero matrix by (6), and also, as the value of γ increases the convergence rate increases, too. The three steps listed below can be used to achieve our goal of developing a ZNN model in line with the Karush–Kuhn–Tucker (KKT) conditions [33,34] to solve the TMCPI problem.

Step 1: **(TMCPI problem reformulation)** The TMCPI problem of (3)–(5) can be reformulated as follows:

$$\min_{\phi(t)} \quad p^{\mathrm{T}}(t) \cdot \phi(t) \tag{7}$$

$$\text{s. t.} \quad -X^{\mathrm{T}}(t) \cdot \phi(t) \leq \min\{-X^{\mathrm{T}}(t) \cdot \theta, -k\} \tag{8}$$

$$\phi(t) \leq \zeta(t) \tag{9}$$

$$-\phi(t) \leq \mathbf{0}, \tag{10}$$

or equivalently

$$\min_{\phi(t)} \quad p^{\mathrm{T}}(t) \cdot \phi(t) \tag{11}$$

$$\text{s. t.} \quad A(t)\phi(t) \leq b(t), \tag{12}$$

where $A(t) = \begin{bmatrix} -X^{\mathrm{T}}(t) \\ I_n \\ -I_n \end{bmatrix} \in \mathbb{R}^{(2n+1)\times n}$ and $b(t) = \begin{bmatrix} \min\{-X^{\mathrm{T}}(t)\cdot\theta, -k\} \\ \zeta(t) \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n+1}$.

Step 2: **(Minimization conditions)** To address the TLP problem of (11)–(12), the following Lagrange function is defined:

$$L(\phi(t), \lambda(t), R(t), t) = p^{\mathrm{T}}(t)\phi(t) + \lambda^{\mathrm{T}}(t)(A(t)\phi(t) + R^{\odot 2}(t) - b(t)), \tag{13}$$

where $\lambda(t) \in \mathbb{R}^{2n+1}$ and $R^{\odot 2}(t) \in \mathbb{R}^{2n+1}$ is a non-negative time-varying term that converts the inequality constraint to an equality constraint. Then, using the KKT conditions [35], we have

$$\begin{cases} \frac{\partial L(\phi(t),\lambda(t),R(t),t)}{\partial \phi(t)} = p(t) + A^{\mathrm{T}}(t)\lambda(t) = 0 \\ \frac{\partial L(\phi(t),\lambda(t),R(t),t)}{\partial \lambda(t)} = A(t)\phi(t) + R^{\odot 2}(t) - b(t) = 0 \\ \frac{\partial L(\phi(t),\lambda(t),R(t),t)}{\partial R(t)} = \lambda(t) \odot R(t) = 0 \end{cases} \tag{14}$$

Step 3: **(ZNN method)** Considering the next equation group of error matrices

$$\begin{cases} K_1(t) = p(t) + A^{\mathrm{T}}(t)\lambda(t) \\ K_2(t) = A(t)\phi(t) + R^{\odot 2}(t) - b(t) \\ K_3(t) = \lambda(t) \odot R(t) \end{cases} \tag{15}$$

and then substituting $K_i(t), i = 1, 2, 3$, defined in (15) in place of $E(t)$ into (6), one obtains

$$\begin{cases} \dot{p}(t) + A^{\mathrm{T}}(t)\dot{\lambda}(t) + \dot{A}^{\mathrm{T}}(t)\lambda(t) = -\gamma K_1(t) \\ \dot{A}(t)\phi(t) + A(t)\dot{\phi}(t) + 2R(t)\dot{R}(t) - \dot{b}(t) = -\gamma K_2(t) \\ \dot{\lambda}(t)R(t) + \lambda(t)\dot{R}(t) = -\gamma K_3(t) \end{cases} \tag{16}$$

where,

$$\dot{A}(t) = \begin{bmatrix} -\dot{X}^{\mathrm{T}}(t) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(2n+1)\times n}, \quad \dot{b}(t) = \begin{bmatrix} \min\{-\dot{X}^{T}(t)\cdot\theta, 0\} \\ \dot{\zeta}(t) \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n+1} \tag{17}$$

Then setting

$$\begin{aligned} S(t) &= \begin{bmatrix} \mathbf{0}_{n\times n} & A^{\mathrm{T}}(t) & \mathbf{0}_{n\times(2n+1)} \\ A(t) & \mathbf{0}_{(2n+1)\times(2n+1)} & 2\bar{R}(t) \\ \mathbf{0}_{(2n+1)\times n} & \bar{R}(t) & \bar{\lambda}(t) \end{bmatrix}, \\ u(t) &= \begin{bmatrix} -\gamma K_1(t) - \dot{p}(t) - \dot{A}^{\mathrm{T}}(t)\lambda(t) \\ -\gamma K_2(t) - \dot{A}(t)x(t) + \dot{b}(t) \\ -\gamma K_3(t) \end{bmatrix}, \end{aligned} \tag{18}$$

where $S(t) \in \mathbb{R}^{(5n+2)\times(5n+2)}$ and $u(t) \in \mathbb{R}^{5n+2}$, (16) can be reformulated as follows:

$$S(t)\dot{y}(t) = u(t), \tag{19}$$

where $\dot{y}(t) = \begin{bmatrix} \dot{\phi}(t) \\ \dot{\lambda}(t) \\ \dot{R}(t) \end{bmatrix} \in \mathbb{R}^{5n+2}$ and $S(t)$ acts as the corresponding mass matrix. The model of (19) is adequate for addressing the TLP problem of (3)–(5). The solution to (19) is given by:

$$\dot{y}(t) = (S^T(t)S(t))^{\dagger}S^T(t)u(t), \tag{20}$$

where $(\cdot)^{\dagger}$ denotes the Moore–Penrose inverse. To establish that the ZNN solver converges to the theoretical solution, we give the following theorem.

Theorem 1. *The state vector* $y(t) = [\phi(t), \lambda(t), R(t)]^{\mathrm{T}}$ *of a ZNN* (20) *converges universally to the theoretical solution* $y^*(t) = [\phi^*(t), \lambda^*(t), R^*(t)]^{\mathrm{T}}$ *starting from any initial condition* $y(0) \in \mathbb{R}^{5n+2}$. *To put it another way,* $\lim_{t\to\infty}(y(t) - y^*(t)) = 0$, *while the first n components of* $y^*(t)$ *are the theoretical solution* $\phi^*(t)$ *of the optimization problem in* (3)–(5).

Proof. The error matrix equation group is determined as in (15), in line with the ZNN architecture, to achieve the solution $\phi(t)$ of TLP (3)–(5). The model (16) is then obtained by using the linear design formula for zeroing (15). When $t \to \infty$, each error matrix equation in the group (16) converges to the zero matrix, according to [14] (Theorem 1). As a consequence, when $t \to \infty$, the solution $y(t)$ of (16) converges to the theoretical solution $y^*(t)$ of TLP (3)–(5). Additionally, we can infer from the (20) derivation procedure that it is merely a variant of the (16) error. Thus, the proof is finished. □

3.2. TMCPI Problem via LVI-PDNN and S-LVI-PDNN

In this section, we discuss two ENN solvers, the LVI-PDNN and the S-LVI-PDNN, both developed by Zhang et al. [36]. The LVI-PDNN may solve online (3)–(5), by setting

$$S(t) = \begin{bmatrix} \mathbf{0}_{n\times n} & -X(t) \\ X^{\mathrm{T}}(t) & 0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}, \quad v(t) = \begin{bmatrix} p(t) \\ \min\{A(t)\cdot\theta, -k\} \end{bmatrix} \in \mathbb{R}^{n+1}, \tag{21}$$

where $w(t)$ is the primal dual-decision vector and the lower and higher boundaries in (5) are as below:

$$w(t)=\begin{bmatrix}\phi(t)\\ c(t)\end{bmatrix}\in\mathbb{R}^{n+1}, \quad \rho^-(t)=\begin{bmatrix}\mathbf{0}\\ 0\end{bmatrix}\in\mathbb{R}^{n+1}, \quad \rho^+(t)=\begin{bmatrix}\zeta(t)\\ 1e{+}100\end{bmatrix}\in\mathbb{R}^{n+1}, \tag{22}$$

The $c(t) \in \mathbb{R}$ denotes the dual-decision vector of inequality constraint (4), while $\phi(t)$ is the decision parameter of (5).

The following dynamical model, known as the LVI-PDNN, can be used to solve the TMCPI problem of (3)–(5):

$$\dot{w}(t) = \gamma(I_{n+1} + S^{\mathrm{T}}(t))(\Psi(w(t) - (S(t)w(t) + v(t))) - w(t)), \tag{23}$$

where $\gamma > 0$ signifies the design parameter and the projection operator is the following element-wise function:

$$\Psi_i(w_i(t)) = \begin{cases} \rho_i^-(t), & w_i(t) < \rho_i^-(t) \\ w_i(t), & \rho_i^-(t) \le w_i(t) \le \rho_i^+(t), \\ \rho_i^+(t), & w_i(t) > \rho_i^+(t) \end{cases} \quad \text{for } i = 1, 2, ..., 2+n. \tag{24}$$

The price of γ must be set to the greatest possible value or selected appropriately within hardware constraints for experimental or simulation purposes. It should be noted that a MATLAB `ode` solver can be used to effectively produce the solution $w(t)$ of (23).

The simplified version of the LVI-PDNN, called the S-LVI-PDNN, is the following dynamical model:

$$\dot{w}(t) = \gamma(\Psi(w(t) - (S(t)w(t) + v(t))) - w(t)), \tag{25}$$

which can also be used to solve the TMCPI problem of (3)–(5).

To better understand the real-time convergence of the LVI-PDNN and S-LVI-PDNN, the residual error is calculated as bellow:

$$K(t) = w(t) - \Psi(w(t) - (S(t)w(t) + v(t))). \tag{26}$$

According to [36] (Theorems 1 and 2) and [37] (Theorem 3), if $\|K(t)\|_2^2 \to 0$, the convergence of the state vector $w(t)$ to the optimal solution $w^*(t)$ can be accomplished, and thus the LVI-PDNN and S-LVI-PDNN converge to the theoretical solution.

4. Real-World Experiments

We use the time series of portfolio returns and insurance prices as input, so the data are in DT. To solve a CT problem and provide an online solution, we must first convert the input data from DT to CT. We accomplish this by employing interpolation functions on $X(t)$, $\dot{X}(t)$, $p(t)$, and $\dot{p}(t)$. It is worth mentioning that the time derivatives of $\zeta(t)$, $X(t)$, and $p(t)$ can be created by setting $\dot{\zeta}(t) = \zeta(t+1) - \zeta(t)$, $\dot{X}(t) = X(t+1) - X(t)$ and $\dot{p}(t) = p(t+1) - p(t)$, respectively. Here, we apply linear interpolation to the DT data we have. To accomplish this, we use the MATLAB custom function `linots`, which is retrieved from [38]. That is, the DT arrays $X(t)$, $\dot{X}(t)$, $p(t)$, and $\dot{p}(t)$ are converted into interpolated CT functions using `linots`. A number of alternative interpolation techniques can be found in [38]. Their major benefit, over commercial MathWorks functions, is that they are designed to reduce the computational costs of the MATLAB `ode` solvers we used. Therefore, when time series constitute the input data, the ENNs produce quicker results.

Moreover, the `ode15s` MATLAB solver is employed in (20), (23), and (25) to generate the online solution of the TMCPI problem with the parameters set to $\gamma = 1e3$, $\beta = 2$, and $\delta = 1e3$. The financial time series are obtained from https://finance.yahoo.com, accessed on 10 August 2022. The following GitHub link leads to the entire implementation and development of the TMCPI problem discussed in Sections 3 and 4: https://github.com/SDMourtas/TMCPI, accessed on 10 August 2022. Lastly, the solutions provided by the ZNN, LVI-PDNN, and S-LVI-PDNN are contrasted to the presumptive theoretical solutions produced by the MATLAB function `linprog`. It is worth mentioning that all numerical experiments are performed using the MATLAB R2021a environment on an Intel® Core™ i5-6600K CPU 3.50 GHz, 16 GB RAM, running on a Windows 10 64-bit Operating System.

Portfolio Cases

Four alternative portfolio setup cases are covered by the trials. In Figure 1, the portfolio cases are shown, with the portfolios' market including stocks that are among the most active on the US market. In the i-th case, $i = 1, 2, 3, 4$, we consider the market $X = [x_1, x_2, \ldots, x_s]$, where X includes the daily close prices of the s stocks shown in Figure 1, for the time period 2 April 2019 to 1 October 2019. It is worth noting that s includes all stocks from the beginning until the case of interest, so case two, for example, includes BAC, MS, F, INTC, and JPM. Setting the time delay parameter $\tau = 21$ for calculating the insurance prices, we try to find through ENNs the optimal portfolio $\phi(t)$ for the time period 1 May 2019 to 1 October 2019. It is important to note that for solving the omitted recorded prices problem between periods of the same division, we utilize the parameter ω from [38], which splits the recorded prices to the time periods for each t inside the ZNN, LVI-PDNN, and S-LVI-PDNN. More precisely, we set

$$\omega = \begin{cases} 22 & , t \in [0,1) \\ (22 + 20 \cdot (t-1))/t & , t \in [1,2) \\ (42 + 22 \cdot (t-2))/t & , t \in [2,4) \\ (86 + 21 \cdot (t-4))/t & , t \in [4,5] \end{cases}$$

which separates the time series from 1 May 2019 to 1 October 2019 into five monthly intervals. Then, we employ $X(\omega t)$ and $p(\omega t)$ instead of $X(t)$ and $p(t)$ and $\dot{X}(\omega t)$ and $\dot{p}(\omega t)$ instead of $\dot{X}(t)$ and $\dot{p}(t)$. As a result, we set $tspan = [0, 5]$ in the MATLAB `ode15s` solver to find the optimal portfolio $\phi(t)$ for the time period 1 May 2019 to 1 October 2019. Given an initial portfolio $\theta = 2\text{ones}(s, 1)$ and a floor k, and starting from $y(0) = [\theta; 2\text{ones}(4s + 2, 1)]$ in (20) and $w(0) = [\theta; 2]$ in (23) and (25), the results are presented in Figures 2 and 3.

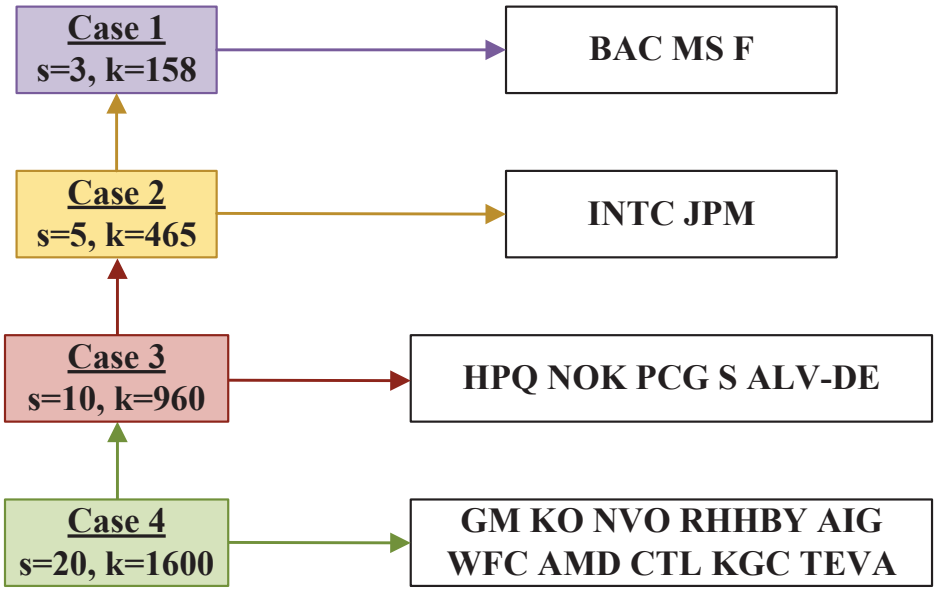


Figure 1. The portfolio cases' stocks that have been utilized in the TMCPI problem experiments.

For the portfolio cases 1 and 2, which deal with three- and five-stock portfolios, respectively, Figure 2a,e show the weights of the optimal insured portfolio $\phi(t)$ generated by the ZNN, LVI-PDNN, S-LVI-PDNN, and `linprog`. In Figure 2a, we can see that all of the portfolios in case 1 are identical with the exception of the portfolio generated by the ZNN, which is slightly different, whereas in Figure 2e, we can see that only the portfolios of the LVI-PDNN and `linprog` in case 2 are identical with the remainder being marginally different. Figure 2b,f show the insurance costs incurred by each portfolio, i.e., the initial portfolio θ and the portfolios $\phi(t)$ generated by the ZNN, LVI-PDNN, S-LVI-PDNN, and `linprog`. We can see that the portfolios in cases 1 and 2 are almost identical. Figure 2c,g show the payoff of each portfolio $\phi(t)$ along with the floor price k. Then, the portfolios in cases 1 and 2 are identical with the exception of the portfolio generated by the S-LVI-PDNN, which is marginally different and begins to become excessively noisy in Figure 2g of case 2. The errors between the solutions of `linprog` and the ENNs, which are presented in Figure 2d,h for cases 1 and 2, respectively, can support the aforementioned. That is, while the S-LVI-PDNN accuracy declines significantly as the problem's dimensions rise, the LVI-PDNN has the same convergence rate and marginally higher accuracy than the ZNN. As a result, the S-LVI-PDNN method has been left out for portfolio cases 3 and 4, which deal with larger portfolios.

In portfolio cases 3 and 4, we deal with 10- and 20-stock portfolios, respectively. The results generated by the ZNN, the LVI-PDNN, and the `linprog` are shown in Figure 3. Figure 3a,d, which correspond to cases 3 and 4, respectively, show the insurance costs incurred by each portfolio. Figure 3b,e, which correspond to cases 3 and 4, respectively, show the payoff of each portfolio along with the floor price k. There, we can see that all portfolios are identical. The errors between the solutions of `linprog` and the ZNN and LVI-PDNN, which are presented in Figure 3c,f for cases 3 and 4, respectively, show that the LVI-PDNN has the same convergence rate and marginally higher accuracy than the ZNN.

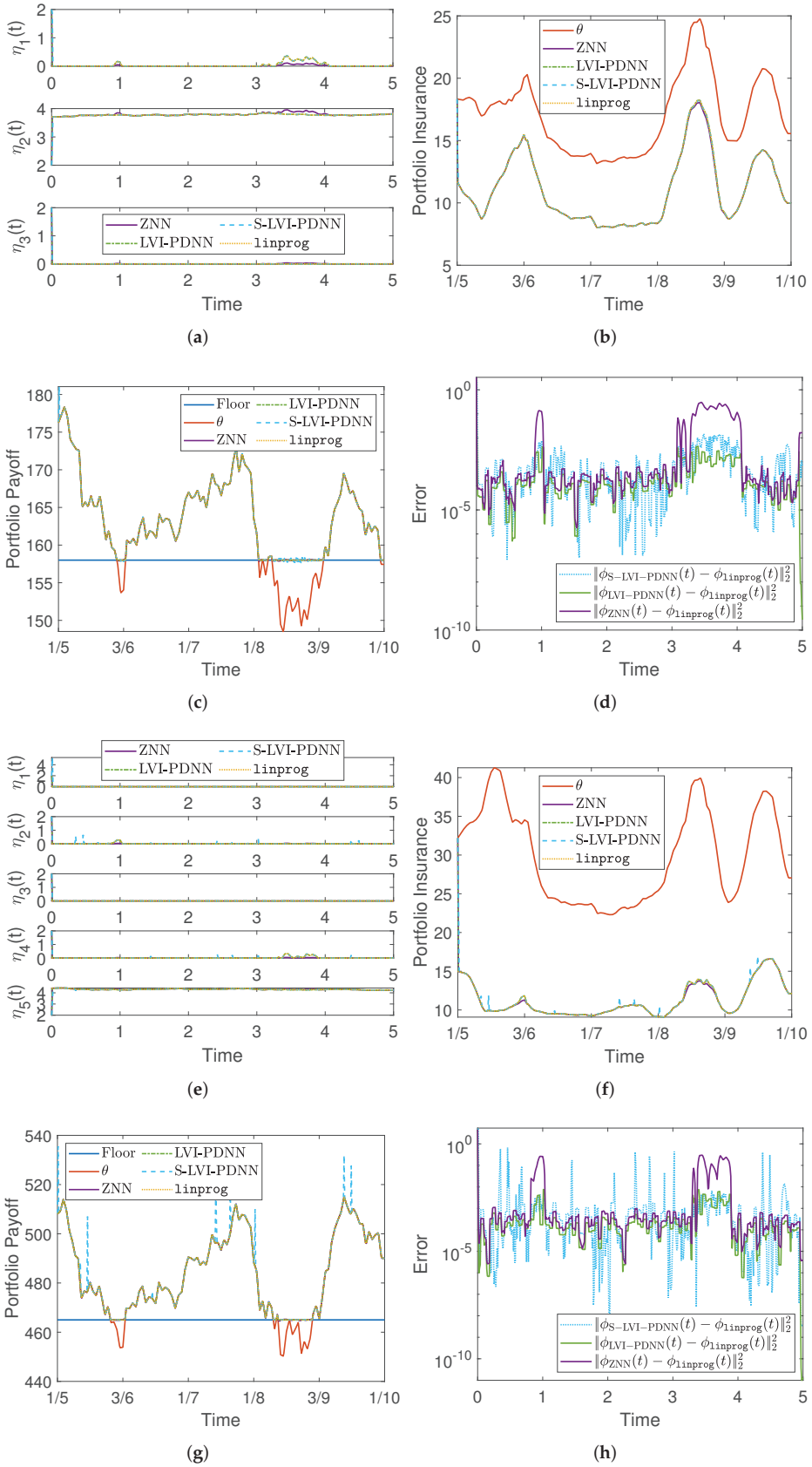


Figure 2. Portfolios' weights, insurance costs, payoff, and the error between NNs and MATLAB's `linprog` in cases 1 and 2. (**a**) Portfolios' weights (case 1); (**b**) insurance costs (case 1); (**c**) portfolios' payoff (case 1).; (**d**) error: NNs vs. `linprog` (case 1); (**e**) portfolios' weights (case 2); (**f**) insurance costs (case 2); (**g**) portfolios' payoff (case 2); (**h**) error: NNs vs. `linprog` (case 2).

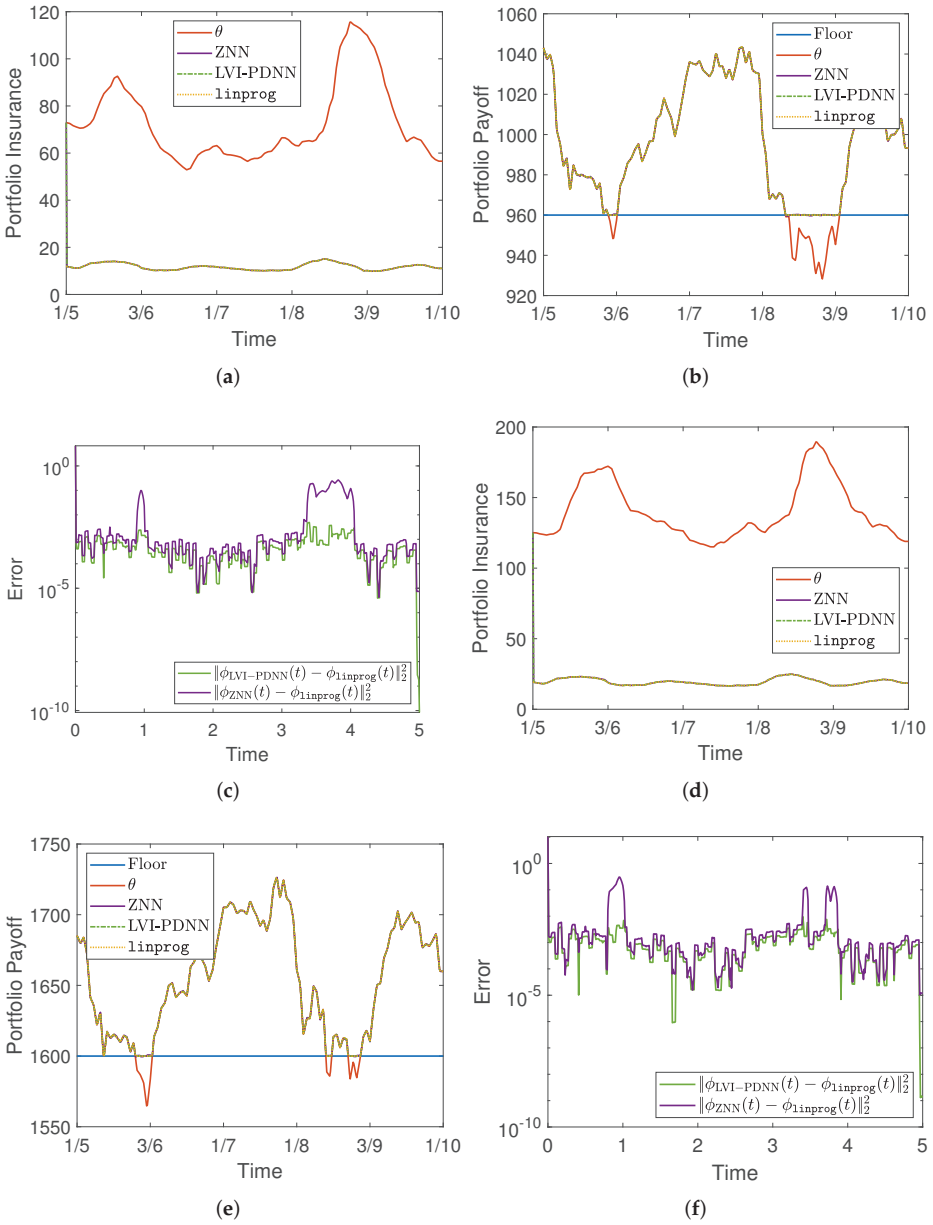


Figure 3. Portfolios' weights, insurance costs, payoff, and the error between NNs and MATLAB's `linprog` in cases 3 and 4. (**a**) Insurance costs (case 3); (**b**) portfolios' payoff (case 3); (**c**) error: NNs vs. `linprog` (case 3); (**d**) insurance costs (case 4); (**e**) portfolios' payoff (case 4); (**f**) error: NNs vs. `linprog` (case 4).

In general, by comparing the insurance costs of Figures 2b,f and 3a,d to the portfolios' payoff in Figures 2c,g and 3b, we can observe that the insurance costs of $\phi(t)$ are increasing only when the payoff must remain at the floor. Additionally, from Figures 2c,g and 3b, it is evident that the payoff of portfolio $\phi(t)$ is greater than the payoff of portfolio θ. Our method also becomes more practical when the ω parameter is considered, which comes in handy when combining time periods that each have a distinct amount of recorded prices. Another major finding is that, in all of the cases, the portfolio insurance costs are significantly lower when we ask for a TV portfolio $\phi(t)$ instead of a constant portfolio θ. It is important to mention that the noise in Figures 2d,h and 3c,f is expected because we are dealing with time-series. Now, by taking into account the design parameter's γ low value, the error value is excellent. Note that the overall error value of the ZNN, the LVI-PDNN, and the S-LVI-PDNN decreases even more when the price of parameter γ

increases. However, we can presume that the LVI-DPNN solver outperforms the ZNN because the error among `linprog` and LVI-DPNN is lower than the error among `linprog` and the ZNN. In addition, since the error among `linprog` and S-LVI-DPNN is more noisy than the error among `linprog` and the ZNN, we can presume that the ZNN performs better than the S-LVI-DPNN. Overall, the portfolio cases presented in the numerical experiments of this section demonstrate that the ZNN, LVI-PDNN, and S-LVI-PDNN work effectively in addressing the TMCPI problem.

Moreover, we provide Table 1 for comparing the performance of various MATLAB custom interpolation functions, which are taken from [38]. Specifically, the functions employed are the `linots`, the `pchinots`, and the `splinots` for linear, P.C. Hermite, and C. Spline interpolation, respectively. This table presents the average consumption time of the ZNN, LVI-PDNN, S-LVI-PDNN, and `linprog` on each portfolio case, by using all the aforementioned MATLAB functions. According to Table 1, P.C. Hermite is the least effective technique, while linear interpolation is the most effective. The fastest ENN for addressing the TMCPI problem is the LVI-PDNN and the slowest is the S-LVI-PDNN. With the exception of case 4, the ZNN and LVI-PDNN outperform `linprog` in all cases when using the linear interpolation method. With the exception of case 3, the LVI-PDNN outperforms the ZNN in all cases when using the linear interpolation method, whereas with the exception of case 1, the ZNN outperforms the LVI-PDNN in all cases when using the C. Spline interpolation method. For all three interpolation methods, only case 1 shows the S-LVI-PDNN to be superior to `linprog`. The performance of ENNs generally declines as the problem's dimension rises, and it is also sensitive to the interpolation method selected. It is important to mention that these MATLAB custom functions, which handle time series, are the best choices in terms of computing time responses, even in the case where they yield identical results as the corresponding conventional MATLAB functions [20]. Based on Table 1 and the analysis presented in this section, it is demonstrated that the ZNN, LVI-PDNN, and S-LVI-PDNN are effective and computationally efficient.

Table 1. The NNs' and MATLAB's solver time consumption for solving the MCPI problem.

Interpolation Method	Case 1				Case 3		
	3 Stocks Portfolio				10 Stocks Portfolio		
	ZNN	LVI-PDNN	S-LVI-PDNN	Linprog	ZNN	LVI-PDNN	`Linprog`
Linear	0.6 s	0.45 s	2.1 s	3.6 s	1.3 s	1.6 s	3.6 s
P.C. Hermite	5.3 s	0.5 s	2.5 s	3.8 s	2.4 s	5.1 s	3.6 s
C. Spline	0.55 s	0.45 s	0.53 s	3.5 s	3.9 s	5.5 s	3.7 s
Interpolation Method	**Case 2**				**Case 4**		
	5 Stocks Portfolio				20 Stocks Portfolio		
	ZNN	LVI-PDNN	S-LVI-PDNN	Linprog	ZNN	LVI-PDNN	`Linprog`
Linear	5.8 s	0.6 s	3.6 s	3.8 s	6.3 s	5.1 s	3.7 s
P.C. Hermite	11.1 s	1.3 s	4.6 s	3.6 s	58 s	12.2 s	4.3 s
C. Spline	1.1 s	1.8 s	5.1 s	3.8 s	6.1 s	10.7 s	3.5 s

5. Conclusions

The TMCPI problem is introduced in this paper, as well as three ENN models, namely the ZNN, LVI-PDNN, and S-LVI-PDNN, for addressing it. The focus of this research is on the use of ENN computational techniques to address the TMCPI problem accurately in a short amount of time. Simulations on portfolios of up to 20 stocks with real-world data have proved that the ENNs are capable of efficiently solving the financial TLP problem, in some cases more than five times faster than traditional approaches. However, their accuracy declines as the size of the portfolio increases. According to the results of the simulations, which included four portfolio configuration cases, the LVI-PDNN was found to be the most efficient solver in addressing the TMCPI problem, whereas the S-LVI-PDNN was found to be the least efficient. A study limitation is the maximum portfolio dimension because the ENNs approach is unable to handle problems with very big dimensions.

Several potential study areas can be investigated:

1. The application of ENNs to higher-dimensional portfolios.

2. The application of ENNs to a variety of financial portfolio optimization problems.
3. The improvement of ENNs' performance in real-world problems by making use of various activation functions.
4. The application of ENNs to portfolios derived from noisy data.

Author Contributions: V.N.K. (Vladislav N. Kovalnogov): methodology, validation, and investigation. R.V.F.: methodology, validation, and investigation. D.A.G.: methodology, validation, and investigation. A.V.C.: methodology, validation, and investigation. V.N.K. (Vasilios N. Katsikis): conceptualization, methodology, validation, formal analysis, investigation, and writing—original draft. S.D.M.: conceptualization, methodology, validation, formal analysis, investigation, and writing—original draft. T.E.S.: methodology, formal analysis, and investigation. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by a Mega Grant from the Government of the Russian Federation within the framework of federal project No. 075-15-2021-584.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NN	neural network
TLP	time-varying linear programming
ENN	error-correction neural network
RNN	recurrent neural network
MCPI	minimum-cost portfolio insurance
TMCPI	time-varying minimum-cost portfolio insurance
ZNN	zeroing neural network
LVI-PDNN	linear-variational-inequality primal-dual neural network
S-LVI-PDNN	simplified linear-variational-inequality primal-dual neural network
TV	time-varying
CT	continuous-time
DT	discrete-time
KKT	Karush–Kuhn–Tucker

References

1. Katsikis, V.N.; Mourtas, S.D. A heuristic process on the existence of positive bases with applications to minimum-cost portfolio insurance in C[a, b]. *Appl. Math. Comput.* **2019**, *349*, 221–244. [CrossRef]
2. Katsikis, V.N.; Mourtas, S.D. ORPIT: A matlab toolbox for option replication and portfolio insurance in incomplete markets. *Comput. Econ.* **2019**, *56*, 711–721. [CrossRef]
3. Medvedeva, M.A.; Katsikis, V.N.; Mourtas, S.D.; Simos, T.E. Randomized time-varying knapsack problems via binary beetle antennae search algorithm: Emphasis on applications in portfolio insurance. *Math. Methods Appl. Sci.* **2020**, *44*, 2002–2012. [CrossRef]
4. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying minimum-cost portfolio insurance under transaction costs problem via Beetle Antennae Search Algorithm (BAS). *Appl. Math. Comput.* **2020**, *385*, 125453. [CrossRef]
5. Katsikis, V.N.; Mourtas, S.D. Portfolio Insurance and Intelligent Algorithms. In *Modeling and Optimization in Science and Technologies*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 18, pp. 305–323. [CrossRef]
6. Simos, T.E.; Mourtas, S.D.; Katsikis, V.N. Time-varying Black-Litterman portfolio optimization using a bio-inspired approach and neuronets. *Appl. Soft Comput.* **2021**, *112*, 107767. [CrossRef]
7. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D. A multi-input with multi-function activated weights and structure determination neuronet for classification problems and applications in firm fraud and loan approval. *Appl. Soft Comput.* **2022**, *127*, 109351. [CrossRef]
8. Leung, M.F.; Wang, J. Minimax and biobjective portfolio selection based on collaborative neurodynamic optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2825–2836. [CrossRef]

9. Yaman, I.; Dalkılıç, T.E. A hybrid approach to cardinality constraint portfolio selection problem based on nonlinear neural network and genetic algorithm. *Expert Syst. Appl.* **2021**, *169*, 114517. [CrossRef]
10. Wang, H.; Zhou, X.Y. Continuous-time mean-variance portfolio selection: A reinforcement learning framework. *Math. Financ.* **2020**, *30*, 1273–1308. [CrossRef]
11. Imajo, K.; Minami, K.; Ito, K.; Nakagawa, K. Deep portfolio optimization via distributional prediction of residual factors. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 35, pp. 213–222. [CrossRef]
12. Zhang, Z.; Zohren, S.; Roberts, S. Deep learning for portfolio optimization. *J. Financ. Data Sci.* **2020**, *2*, 8–20. [CrossRef]
13. Ma, Y.; Han, R.; Wang, W. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Syst. Appl.* **2021**, *165*, 113973. [CrossRef]
14. Zhang, Y.; Ge, S.S. Design and analysis of a general recurrent neural network model for time-varying matrix inversion. *IEEE Trans. Neural Netw.* **2005**, *16*, 1477–1490. [CrossRef]
15. Kornilova, M.; Kovalnogov, V.; Fedorov, R.; Zamaleev, M.; Katsikis, V.N.; Mourtas, S.D.; Simos, T.E. Zeroing neural network for pseudoinversion of an arbitrary time-varying matrix based on singular value decomposition. *Mathematics* **2022**, *10*, 1208. [CrossRef]
16. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Gerontitis, D. A higher-order zeroing neural network for pseudoinversion of an arbitrary time-varying matrix with applications to mobile object localization. *Inf. Sci.* **2022**, *600*, 226–238. [CrossRef]
17. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Zhang, Y. Solving complex-valued time-varying linear matrix equations via QR decomposition with applications to robotic motion tracking and on angle-of-arrival localization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–10. [CrossRef]
18. Jiang, W.; Lin, C.L.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Simos, T.E. Zeroing neural network approaches based on direct and indirect methods for solving the Yang–Baxter-like matrix equation. *Mathematics* **2022**, *10*, 1950. [CrossRef]
19. Mourtas, S.D.; Katsikis, V.N. Exploiting the Black-Litterman framework through error-correction neural networks. *Neurocomputing* **2022**, *498*, 43–58. [CrossRef]
20. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying mean-variance portfolio selection problem solving via LVI-PDNN. *Comput. Oper. Res.* **2022**, *138*, 105582. [CrossRef]
21. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Zhang, Y. Continuous-time varying complex QR decomposition via zeroing neural dynamics. *Neural Process. Lett.* **2021**, *53*, 3573–3590. [CrossRef]
22. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S. Unique non-negative definite solution of the time-varying algebraic Riccati equations with applications to stabilization of LTV systems. *Math. Comput. Simul.* **2022**, *202*, 164–180. [CrossRef]
23. Ke, F.; Li, Z.; Yang, C. Robust tube-based predictive control for visual servoing of constrained differential-drive mobile robots. *IEEE Trans. Ind. Electron.* **2018**, *65*, 3437–3446. [CrossRef]
24. Jin, L.; Zhang, Y.; Li, S.; Zhang, Y. Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators. *IEEE Trans. Ind. Electron.* **2016**, *63*, 6978–6988. [CrossRef]
25. Annaert, J.; Ceuster, M.D.; Vandenbroucke, J. Mind the floor: Enhance portfolio insurance without borrowing. *J. Investig.* **2019**, *28*, 39–50. [CrossRef]
26. Matsumoto, K. Portfolio insurance with liquidity risk. *Asia-Pac. Financ. Mark.* **2008**, *14*, 363. [CrossRef]
27. Aliprantis, C.D.; Brown, D.J.; Werner, J. Minimum-cost portfolio insurance. *J. Econ. Dyn. Control.* **2000**, *24*, 1703–1719. [CrossRef]
28. Zhang, Y.; Guo, D. Linear programming versus quadratic programming in robots' repetitive redundancy resolution: A chattering phenomenon investigation. In Proceedings of the 4th IEEE Conference Industrial Electronics and Applications, Xi'an, China, 25–27 May 2009; pp. 2822–2827. [CrossRef]
29. Zhang, Y.; Ma, W.; Li, X.; Tan, H.; Chen, K. MATLAB Simulink modeling and simulation of LVI-based primal-dual neural network for solving linear and quadratic programs. *Neurocomputing* **2009**, *72*, 1679–1687. [CrossRef]
30. Zhang, Y.; Leithead, W.E. Exploiting Hessian matrix and trust-region algorithm in hyperparameters estimation of Gaussian process. *Appl. Math. Comput.* **2005**, *171*, 1264–1281. [CrossRef]
31. Zhang, Y. On the LVI-based primal-dual neural network for solving online linear and quadratic programming problems. In Proceedings of the American Control Conference, Portland, OR, USA, 8–10 June 2005; Volume 2, pp. 1351–1356. [CrossRef]
32. Zhang, Y.; Wang, J. Recurrent neural networks for nonlinear output regulation. *Automatica* **2001**, *37*, 1161–1173. [CrossRef]
33. Jin, L.; Li, S. Nonconvex function activated zeroing neural network models for dynamic quadratic programming subject to equality and inequality constraints. *Neurocomputing* **2017**, *267*, 107–113. [CrossRef]
34. Li, J.; Shi, Y.; Xuan, H. Unified model solving nine types of time-varying problems in the frame of zeroing neural network. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 1896–1905. [CrossRef]
35. Boyd, S.; Vandenberghe, L. *Convex Optimization Problems*; Cambridge University Press: New York, NY, USA, 2004; pp. 127–214. [CrossRef]
36. Zhang, Y.; Wu, F.; Xiao, Z.; Li, Z.; Cai, B. Performance analysis of LVI-based PDNN applied to real-time solution of time-varying quadratic programming. In Proceedings of the 2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, 6–11 July 2014; pp. 3155–3160. [CrossRef]

37. Zhang, Y.; Li, Z.; Tan, H.Z.; Fan, Z. On the simplified lvi-based primal-dual neural network for solving LP and QP problems. In Proceedings of the 2007 IEEE International Conference on Control and Automation, Guangzhou, China, 30 May–1 June 2007. [CrossRef]
38. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying mean-variance portfolio selection under transaction costs and cardinality constraint problem via beetle antennae search algorithm (BAS). *SN Oper. Res. Forum* **2021**, *2*, 18. [CrossRef]

mathematics

MDPI

Article

Exploiting Mean-Variance Portfolio Optimization Problems through Zeroing Neural Networks

Spyridon D. Mourtas [1] and Chrysostomos Kasimis [2,*]

[1] Department of Economics, Division of Mathematics and Informatics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece
[2] Department of Physics, Electronics Laboratory, University of Patras, 26504 Patras, Greece
* Correspondence: chrkasim@upatras.gr

Abstract: In this research, three different time-varying mean-variance portfolio optimization (MVPO) problems are addressed using the zeroing neural network (ZNN) approach. The first two MVPO problems are defined as time-varying quadratic programming (TVQP) problems, while the third MVPO problem is defined as a time-varying nonlinear programming (TVNLP) problem. Then, utilizing real-world datasets, the time-varying MVPO problems are addressed by this alternative neural network (NN) solver and conventional MATLAB solvers, and their performances are compared in three various portfolio configurations. The results of the experiments show that the ZNN approach is a magnificent alternative to the conventional methods. To publicize and explore the findings of this study, a MATLAB repository has been established and is freely available on GitHub for any user who is interested.

Keywords: Markowitz framework; mean-variance portfolio optimization (MVPO); zeroing neural network (ZNN); time-varying quadratic programming; time-varying nonlinear programming

MSC: 68T05; 90C20; 91B28

1. Introduction and Motivation

Portfolio management refers to the process of controlling an asset portfolio to achieve an investor's risk tolerance and long-term financial aims. Option replication [1], risk management [2], transaction costs [3], insurance costs [4], liquidity risk [5], and other disciplines of portfolio optimization may be effectively approached using conventional optimization methods. Most recently, neural network (NN) methods such as weights-and-structure-determination (WASD)-based NNs [6], collaborative neurodynamic optimization [7], nonlinear NNs [8], and reinforcement learning [9] have been utilized to address portfolio optimization. This work defines and explores the continuous-time (CT) version of three variations of the mean-variance portfolio optimization (MVPO) problem, while the zeroing NN (ZNN) method is used to solve these financial CT problems. The first two MVPO problems are defined as time-varying quadratic programming (TVQP) problems, while the third MVPO problem is defined as a time-varying nonlinear programming (TVNLP) problem. This article's primary goal is to solve the time-varying MVPO problem through the ZNN method accurately in a short amount of time. The CT versions of the MVPO problems permit the appliance of the ZNN method to the field of finance.

The ZNN framework was created by Zhang et al. in [10] for generating online solutions to TV problems and is based on the Hopfield neural network. Notice that the vast majority of ZNN-based dynamical systems are classified as recurrent NNs (RNNs), which are utilized to locate equations' zeros. The ZNN approach has been widely used to solve a number of TV issues as a result of its thorough examination, with the most common applications being problems of generalized inversion [11], matrix equations systems [12], problems of quadratic optimization [13], linear equations systems [14], and various matrix

functions approximation, such as performing TV QR decomposition [15] and solving TV algebraic Riccati equations [16].

The main benefits of an artificial NN include generalization, fault and noise tolerance, and the capacity to predict data that have not yet been seen while saving costs and time [15]. Based on this and the appeal of utilizing NNs to tackle portfolio selection problems in the recent past, our Markowitz's mean-variance framework approaches take advantage of the benefits of the NN solver (i.e., ZNN method) through the time-varying MVPO problems, which are extremely realistic problems of financial risk management. Additionally, the NN solver may be regarded as predictive dynamics, as is well known. By employing this NN method, the models proposed in [6,11] have magnificent convergence performance, while the convergence speed of the approaches can be changed by altering the design parameter [15]. As a consequence, by using the ZNN method, the time-varying MVPO problems can be addressed with exponential convergence performance. It is worth mentioning that conventional optimization algorithms, such as standard solvers [17], evolutionary algorithms [18], and genetic algorithms [19,20], are typically sufficient for solving the static MVPO. However, conventional optimization methods are only able to solve the time-varying MVPO in the discrete-time (DT) case, whereas the ZNN approach can solve it in both the DT and CT cases. In [21], the authors contended that "Static-time and time-varying problems sometimes behave differently. Therefore time-invariant and time-varying problems may require different approaches." To be able to track the evolution of the static MVPO problems over time and to offer a form of prediction, we investigate the MVPO problems as continuous TVQP and TVNLP problems, respectively. Last but not least, well-known methods for addressing a series of static problems, such as ZNN, exceed the methodology for dealing with time-varying situations.

The main contributions of this work can be summed up as follows:

- Three time-varying MVPO problems are defined and explored;
- Three novel ZNN models for addressing the time-varying MVPO problems are defined;
- For the first time, the ZNN approach has been used to solve a TVNLP problem;
- Using real-world datasets to apply in the field of finance the NN solver;
- The performances of the NN solver and conventional MATLAB solvers are demonstrated and contrasted in trials using three different portfolio configurations.

The following hierarchy governs the overall organization of parts in the document. The three variations of the MVPO problem are introduced in Section 2. Section 3 presents the ZNN solver. Experiments in Section 4 look at the performance and efficacy of the NN solver for resolving the time-varying MVPO problems in three different portfolio setup cases using daily real-world data. Additionally, information about a publicly accessible MATLAB repository on GitHub is provided in Section 4. This repository implements all of the techniques and procedures outlined in Sections 3 and 4 to promote the readability and computational value of this research. The final remarks are found in Section 5.

We will utilize these symbols in the follow-up: $\mathbf{1}$ and $\mathbf{0}$, respectively, for those elements of $\mathbb{R}^n$ consisting of ones and zeros; O_k, I_k, respectively, for the zero and the identity $k \times k$ matrix; $(\cdot)^{\mathrm{T}}$, $(\cdot)^{-1}$, respectively, denote matrix transposition and inversion.

2. Mean-Variance Portfolio Optimization

A portfolio, in finance, is a compilation of all the assets possessed by a public or private institution. In Markowitz's modern portfolio theory (MPT), which was established half a century ago [22], we face the challenge of assigning the funds to the assets that are available in a way that risk decreases and profit increases, where the profit refers to the expected mean return of the portfolio, and the risk refers to the portfolio's variance. That is, the less variance there is, the lesser the risk. Furthermore, short sales are outlawed in the MPT's ideal market, in which shares are endlessly separable and thus could be sold in any (non-negative) portion, free of taxes and transaction costs. This work also adheres to these assumptions.

The MVPO problem is important both practically and theoretically [22]. The MVPO is an optimization problem, in finance, that includes allocating the assets in the portfolio in a way that reduces risk while achieving a desired expected return. As seen in [23–26], the MVPO has been extensively investigated over the past few decades. For instance, different approaches for solving variations of the static MVPO problem are examined in [23,24], a CT MVPO problem with stochastic parameters under a no-bankruptcy limit is investigated in [25], and a multi-period modified MVPO problem is presented and examined in [26]. The definition of the three MVPO problem's variations are covered in great detail in this section.

Assume the marketed space $Q(t) = [q_1(t), q_2(t), \ldots, q_n(t)] \in \mathbb{R}^n$ at time $t = 1, 2, \ldots, \mu$, where $q_i(t) \in \mathbb{R}$ denotes the asset's i, $i = 1, 2, \ldots, n$, return. Assuming the past values (or delays) number $\tau \in \mathbb{N}$, we set $p(t) = [p_1(t), p_2(t), \ldots, p_n(t)] \in \mathbb{R}^n$ the market's space expected return at time t, where $p_i(t) = \sum_{d=0}^{\tau-1} q_i(t-d)/\tau \in \mathbb{R}$ denotes the asset's i, $i = 1, 2, \ldots, n$, expected return. In this way, $p_i(t)$ turns into a weight-free average of the previous τ values, i.e., a simple moving average (SMA) [27]. It is worth mentioning that one of the most frequently utilized technical indicators is the moving average. Furthermore, $C(t) \in \mathbb{R}^{n \times n}$ is covariance matrix of the marketed space and $\eta(t) = [\eta_1(t), \eta_2(t), \ldots, \eta_n(t)] \in \mathbb{R}^n$ is the optimal portfolio. Based on [13,28], the three variations of the MVPO problem are described in the following subsections.

2.1. Time-Varying MVPO Problem (Version 1)

Considering $p_g(t) \in [\min(p(t)), \max(p(t))] \subseteq \mathbb{R}$ the target expected return of the portfolio, the TVQP formulation for the first version of the time-varying MVPO (MVPO1) problem is as follows:

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t) C(t) \eta(t) \tag{1}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}} \eta(t) = 1 \tag{2}$$

$$-p^{\mathrm{T}}(t) \eta(t) \leq -p_g(t) \tag{3}$$

$$\mathbf{0} \leq \eta(t) \leq \mathbf{1}. \tag{4}$$

The constraint in (2) is the typical holding constraint, which requires that the sum of all asset weights be 1, and the constraint in (4) indicates the lower and upper limits of the portfolio asset weights. The constraint in (3) shows that the expected return must equal or exceed the target value $p_g(t)$, whereas the objective function (1) and the portfolio's overall variance are the same. As a result, the MVPO1 problem finds the minimum risk portfolio with an expected return greater or equal to the target $p_g(t)$. That is, solving the MVPO1 problem for values of $p_g(t) \in [\min(p(t)), \max(p(t))]$ one obtains all efficient portfolios.

2.2. Time-Varying MVPO Problem (Version 2)

In this version, $p_g(t) \geq 0$ stands for a risk tolerance factor. A value of $p_g(t) = 0$ indicates a portfolio with minimum risk, while a value of $p_g(t) \to \infty$ indicates a portfolio that is indefinitely out on the frontier with both unbounded expected return and risk. Based on this, the TVQP formulation for the second version of the time-varying MVPO (MVPO2) problem is as follows:

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t) C(t) \eta(t) - p_g(t) \left(p^{\mathrm{T}}(t) \eta(t) \right) \tag{5}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}} \eta(t) = 1 \tag{6}$$

$$\mathbf{0} \leq \eta(t) \leq \mathbf{1}. \tag{7}$$

The constraint in (6) is the typical holding constraint, which requires that the sum of all asset weights be 1, and the constraint in (7) indicates the lower and upper limits of the portfolio asset weights. It is important to mention that the objective function (5) is a risk-adjusted

return function where the constant $p_g(t)$ serves as a risk-aversion constant. As a result, the MVPO2 problem determines the location on the frontier where the inverse of the frontier's slope would be $p_g(t)$.

2.3. Time-Varying MVPO Problem (Version 3)

In this version, $p_g(t) \geq 0$ is a given upper limit on the variance of the portfolio. Based on this, the TVNLP formulation for the third version of the time-varying MVPO (MVPO3) problem is as follows:

$$\min_{\eta(t)} \quad -p^{\mathrm{T}}(t)\eta(t) \tag{8}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{9}$$

$$\eta^{\mathrm{T}}(t)C(t)\eta(t) \leq p_g(t), \tag{10}$$

$$\mathbf{0} \leq \eta(t) \leq \mathbf{1}. \tag{11}$$

The constraint in (9) is the typical holding constraint, which requires that the sum of all asset weights be 1, and the constraint in (11) indicates the lower and upper limits of the portfolio asset weights, whereas the objective function (8) is the portfolio's expected return. Notice that the MVPO3 problem is not a TVQP problem since it has a convex quadratic constraint in (10). As a result, the MVPO3 problem finds the maximum expected return portfolio with a variance below the limit $p_g(t)$.

2.4. Conversion from Discrete-Time to Continuous-Time MVPO Problems

By interpolating the $p(t)$ and the $C(t)$ into continuous functions with any methodology of preferences, we transform the time-varying MVPO problems from DT to CT. As a result, considering the space of all continuous real functions $C[0, \mu - \tau - 1]$ on the interval $[0, \mu - \tau - 1]$, we have that $p(t),\ C(t) \in C[0, \mu - \tau - 1]$, and $\eta(t)$ becomes the online solution of the MVPO1 problem of (1)–(4), the MVPO2 problem of (5)–(7), and the MVPO3 problem of (8)–(11).

3. The Neural Network Approach

Using NNs to address intractability problems and solve complex computation equations is now commonplace in academia and industry. Due to their central significance in mathematical optimization, TVQP problems have gotten a lot of attention in recent decades [29–31], whereas the NN concept is regarded a powerful tool for real-time computation because of its hardware implementation availability and parallel distributed computing nature [15]. TVQP and TVNLP problems may be stated as a set of error equations in the case of RNNs by finding their zeros. ZNN is a type of NNs that is specifically designed to zeroing equations, which has played an important role in the online solution of time-varying problems in recent years by tackling a variety of difficult problems in a variety of scientific domains [13,32]. This section describes the NN solver for approaching the MVPO1, MVPO2, and MVPO3 problems.

3.1. ZNN Approach on the MVPO1 Problem

Since its introduction by Zhang et al. in 2001 [33], the ZNN evolution has been studied and developed as a significant class of recurrent NNs. Furthermore, ZNN has been theoretically investigated and shown to be a powerful and trustworthy method for resolving time-varying problems. The error matrix $Z(t)$ is created and then may be dynamically controlled using the next formula, according to the ZNN design with the linear activation function [12]:

$$\dot{Z}(t) = -\gamma Z(t), \tag{12}$$

where $\gamma > 0$ is the design parameter, and $\dot{(\ )}$ signifies the time derivative. $Z(t)$ is forced to exponentially converge to zero matrix by (12), while it is demonstrated that the convergence

rate increases with the value supplied to γ. The three steps listed below can be used to achieve our goal of developing a ZNN model based on [30,31] in order to solve the MVPO1 problem.

Step 1: **(MVPO1 problem reformulation)** The MVPO1 problem of (1)–(4) can be reformulated as follows:

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t)C(t)\eta(t) \tag{13}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{14}$$

$$-p(t)^{\mathrm{T}}\eta(t) \leq -p_g(t) \tag{15}$$

$$\eta(t) \leq \mathbf{1}, \tag{16}$$

$$-\eta(t) \leq \mathbf{0}, \tag{17}$$

or equivalent

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t)C(t)\eta(t) \tag{18}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{19}$$

$$A(t)\eta(t) \leq b(t), \tag{20}$$

where $A(t) = \begin{bmatrix} -p^{\mathrm{T}}(t) \\ I_n \\ -I_n \end{bmatrix} \in \mathbb{R}^{(1+2n)\times n}$ and $b(t) = \begin{bmatrix} -p_g(t) \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{1+2n}$. Then, using the penalty function proposed in [30,31], the problem of (18)–(20) may be reformulated as below:

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t)C(t)\eta(t) + P(\eta(t)) \tag{21}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1, \tag{22}$$

where $P(\eta(t)) = h\sum_{i=1}^{1+2n} e^{-sN_i(t)} \in \mathbb{R}$ with $N_i(t) = b_i(t) - A_i(t)\eta(t)$, $i = 1, 2, \ldots, 1+2n$. Notice that $h \geq 0$ and $s > 0$, respectively, denote the penalty and the design parameter. The $P(\eta(t))$ value is more consequent with the conditions (20) as s increases, although calculations will take longer if is too large. When $N(t) = 0$, h can push the $P(\eta(t))$ value to be close to zero.

Step 2: **(Conditions of minimization)** The optimization problem in (21) and (22) is solved by determining the following Lagrange function:

$$L(\eta(t), \lambda(t), t) = \eta^{\mathrm{T}}(t)C(t)\eta(t) + P(\eta(t)) + \lambda^{\mathrm{T}}(t)(\mathbf{1}^{\mathrm{T}}\eta(t) - 1). \tag{23}$$

The following are the first-order conditions:

$$\begin{cases} \frac{\partial L(\eta(t),\lambda(t),t)}{\partial \eta(t)} = 2C(t)\eta(t) + P_\eta(t) + \lambda(t)\mathbf{1} = 0 \\ \frac{\partial L(\eta(t),\lambda(t),t)}{\partial \lambda(t)} = \mathbf{1}^{\mathrm{T}}\eta(t) - 1 = 0 \end{cases} \tag{24}$$

where $P_\eta(t) = hs\sum_{i=1}^{1+2n}\left(e^{-sN_i(t)}A_i^{\mathrm{T}}(t)\right) \in \mathbb{R}^n$.

Step 3: **(ZNN solver)** The next error matrix equation group is set:

$$\begin{cases} Z_1(t) = 2C(t)\eta(t) + P_\eta(t) + \lambda(t)\mathbf{1} \\ Z_2(t) = \mathbf{1}^{\mathrm{T}}\eta(t) - 1 \end{cases} \tag{25}$$

and then replacing $Z(t)$ in (12) with $Z_i(t), i = 1, 2$, defined in (25), one obtains:

$$\begin{cases} 2\dot{C}(t)\eta(t) + 2C(t)\dot{\eta}(t) + \dot{P}_\eta(t) + \dot{\lambda}(t)\mathbf{1} + \lambda(t)\mathbf{0} = -\gamma Z_1(t) \\ \mathbf{0}^{\mathrm{T}}\eta(t) + \mathbf{1}^{\mathrm{T}}\dot{\eta}(t) = -\gamma Z_2(t) \end{cases} \tag{26}$$

where $\dot{P}_\eta(t) = P_1(t)\dot{\eta}(t) + P_2(t)\eta(t) + P_3(t)$ with

$$\begin{aligned} P_1(t) &= hs^2 \sum_{i=1}^{1+2n} \left(e^{-sN_i(t)} A_i^{\mathrm{T}}(t) A_i(t) \right) \in \mathbb{R}^{n\times n}, \\ P_2(t) &= hs^2 \sum_{i=1}^{1+2n} \left(e^{-sN_i(t)} A_i^{\mathrm{T}}(t) \dot{A}_i(t) \right) \in \mathbb{R}^{n\times n}, \\ P_3(t) &= hs \sum_{i=1}^{1+2n} \left(e^{-sN_i(t)} \left(\dot{A}_i^{\mathrm{T}}(t) - sA_i^{\mathrm{T}}(t)\dot{b}(t) \right) \right) \in \mathbb{R}^{n}. \end{aligned} \tag{27}$$

It is important to mention that $\dot{A}(t) = \begin{bmatrix} -\dot{p}(t)^{\mathrm{T}} \\ O_n \\ O_n \end{bmatrix} \in \mathbb{R}^{(1+2n)\times n}$ and $\dot{b}(t) = \begin{bmatrix} -\dot{p}_g(t) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{1+2n}$. As a consequence, (26) may be reformulated as below:

$$\begin{cases} (2C(t) + P_1(t))\dot{\eta}(t) + \dot{\lambda}(t)\mathbf{1} = -\gamma Z_1(t) - 2\dot{C}(t)\eta(t) - P_2(t)\eta(t) - P_3(t) \\ \mathbf{1}^{\mathrm{T}}\dot{\eta}(t) = -\gamma Z_2(t) \end{cases} \tag{28}$$

Then setting

$$M(t) = \begin{bmatrix} 2C(t)+P_1(t) & \mathbf{1} \\ \mathbf{1}^{\mathrm{T}} & 0 \end{bmatrix}, \quad v(t) = \begin{bmatrix} -\gamma Z_1(t) - 2\dot{C}(t)\eta(t) - P_2(t)\eta(t) - P_3(t) \\ -\gamma Z_2(t) \end{bmatrix}, \tag{29}$$

where $M(t) \in \mathbb{R}^{(1+n)\times(1+n)}$ and $v(t) \in \mathbb{R}^{1+n}$, (28) may be reformulated as below:

$$M(t)\dot{w}(t) = v(t), \tag{30}$$

where $\dot{w}(t) = \begin{bmatrix} \dot{\eta}(t) \\ \dot{\lambda}(t) \end{bmatrix} \in \mathbb{R}^{1+n}$. Since $M(t)$ is a nonsingular mass matrix, the model of (30) is adequate for addressing the MVPO1 problem of (1)–(4); however, as will be demonstrated in Section 4, since the construction of p and C in our approach in terms of MATLAB code relies on function handle, the next model of (31) is simpler to control under a MATLAB `ode` solver. As a consequence, the following is the suggested ZNN model for the MVPO1 problem of (1)–(4):

$$\dot{w}(t) = M^{-1}(t)v(t), \tag{31}$$

which, with the exception of the expense of computing the matrix's inverse, is equivalent to (30). It is important to note that using a MATLAB solver for the `ode` can effectively produce the solution $w(t)$ of Equation (31). The following theorem, 1, establishes that the ZNN solver converges to the theoretical solution.

Theorem 1. *The state vector $w(t) = [\eta^{\mathrm{T}}(t), \lambda^{\mathrm{T}}(t)]^{\mathrm{T}}$ of ZNN* (31) *converges universally to the theoretical solution $w^*(t) = [\eta^{*\mathrm{T}}(t), \lambda^{*\mathrm{T}}(t)]^{\mathrm{T}}$ starting from any initial condition $w(0) \in \mathbb{R}^{n+k}$. To put it another way, $\lim_{t\to\infty}(w^*(t) - w(t)) = 0$, while the first n components of $x^*(t)$ are the theoretical solution $\eta^*(t)$ of TVQP* (21) *and* (22).

Proof. The error matrix equation group is determined as in (25), inline with the ZNN architecture, to achieve the solution $w(t)$ of TVQP (21) and (22). The model (26) is then obtained by using the linear design formula for zeroing (25). When $t \to \infty$, each error

matrix equation in the group (26) converges to zero matrix, according to ([10] Theorem 1). As a consequence, when $t \to \infty$, the solution of (26) converges to the theoretical solution of TVQP (21) and (22). Additionally, we can infer from the (31) derivation procedure that it is merely a variant of the (26) error. Thus, the proof is finished. □

3.2. ZNN Approach on the MVPO2 Problem

Similar to the ZNN approach on the MVPO1 problem, the three steps listed below can be used to develop a ZNN model in order to solve the MVPO2 problem.

Step 1: **(MVPO2 problem reformulation)** The MVPO2 problem of (5)–(7) can be reformulated as follows:

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t)C(t)\eta(t) - p_g(t)\Big(p(t)^{\mathrm{T}}\eta(t)\Big) \tag{32}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{33}$$

$$\eta(t) \leq \mathbf{1} \tag{34}$$

$$-\eta(t) \leq \mathbf{0}, \tag{35}$$

or equivalent

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t)C(t)\eta(t) - p_g(t)\Big(p(t)^{\mathrm{T}}\eta(t)\Big) \tag{36}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{37}$$

$$A(t)\eta(t) \leq b(t), \tag{38}$$

where $A(t) = \begin{bmatrix} I_n \\ -I_n \end{bmatrix} \in \mathbb{R}^{2n\times n}$ and $b(t) = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n}$. Then, using the penalty function proposed in [30,31], the problem of (36)–(38) may be reformulated as below:

$$\min_{\eta(t)} \quad \eta^{\mathrm{T}}(t)C(t)\eta(t) - p_g(t)(p(t)^{\mathrm{T}}\eta(t)) + P(\eta(t)) \tag{39}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1, \tag{40}$$

where $P(\eta(t)) = h\sum_{i=1}^{2n} e^{-sN_i(t)} \in \mathbb{R}$ with $N_i(t) = b_i(t) - A_i(t)\eta(t)$, $i = 1, 2, \ldots, 2n$. Notice that $h \geq 0$ and $s > 0$, respectively, denote the penalty and the design parameter. The $P(\eta(t))$ value is more consequent with the conditions (38) as s increases, although calculations will take longer if is too large. When $N(t) = 0$, h can push the $P(\eta(t))$ value to be close to zero.

Step 2: **(Conditions of minimization)** The optimization problem in (39) and (40) is solved by determining the following Lagrange function:

$$L(\eta(t), \lambda(t), t) = \eta^{\mathrm{T}}(t)C(t)\eta(t) - p_g(t)(p(t)^{\mathrm{T}}\eta(t)) + P(\eta(t)) + \lambda^{\mathrm{T}}(t)(\mathbf{1}^{\mathrm{T}}\eta(t) - 1). \tag{41}$$

The following are the first-order conditions:

$$\begin{cases} \frac{\partial L(\eta(t),\lambda(t),t)}{\partial \eta(t)} = 2C(t)\eta(t) - p_g(t)p(t) + P_\eta(t) + \lambda(t)\mathbf{1} = 0 \\ \frac{\partial L(\eta(t),\lambda(t),t)}{\partial \lambda(t)} = \mathbf{1}^{\mathrm{T}}\eta(t) - 1 = 0 \end{cases} \tag{42}$$

where $P_\eta(t) = hs\sum_{i=1}^{2n}\Big(e^{-sN_i(t)}A_i^{\mathrm{T}}(t)\Big) \in \mathbb{R}^n$.

Step 3: **(ZNN solver)** The next error matrix equation group is set:

$$\begin{cases} Z_1(t) = 2C(t)\eta(t) - p_g(t)p(t) + P_\eta(t) + \lambda(t)\mathbf{1} \\ Z_2(t) = \mathbf{1}^{\mathrm{T}}\eta(t) - 1 \end{cases} \tag{43}$$

and then replacing $Z(t)$ in (12) with $Z_i(t), i = 1, 2$, defined in (43), one obtains:

$$\begin{cases} 2\dot{C}(t)\eta(t) + 2C(t)\dot{\eta}(t) - \dot{p}_g(t)p(t) - p_g(t)\dot{p}(t) + \dot{P}_\eta(t) + \dot{\lambda}(t)\mathbf{1} + \lambda(t)\mathbf{0} = -\gamma Z_1(t) \\ \mathbf{0}^{\mathrm{T}}\eta(t) + \mathbf{1}^{\mathrm{T}}\dot{\eta}(t) = -\gamma Z_2(t) \end{cases} \tag{44}$$

where $\dot{P}_\eta(t) = P_1(t)\dot{\eta}(t) + P_2(t)\eta(t) + P_3(t)$ with

$$\begin{aligned} P_1(t) &= hs^2 \sum_{i=1}^{2n} \left(e^{-sN_i(t)} A_i^{\mathrm{T}}(t) A_i(t) \right) \in \mathbb{R}^{n\times n}, \\ P_2(t) &= hs^2 \sum_{i=1}^{2n} \left(e^{-sN_i(t)} A_i^{\mathrm{T}}(t) \dot{A}_i(t) \right) \in \mathbb{R}^{n\times n}, \\ P_3(t) &= hs \sum_{i=1}^{2n} \left(e^{-sN_i(t)} \left(\dot{A}_i^{\mathrm{T}}(t) - sA_i^{\mathrm{T}}(t)\dot{b}(t) \right) \right) \in \mathbb{R}^{n}. \end{aligned} \tag{45}$$

It is important to mention that $\dot{A}(t) = \begin{bmatrix} O_n \\ O_n \end{bmatrix} \in \mathbb{R}^{2n\times n}$ and $\dot{b}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n}$. As a consequence, (26) may be reformulated as below:

$$\begin{cases} (2C(t) + P_1(t))\dot{\eta}(t) + \dot{\lambda}(t)\mathbf{1} = -\gamma Z_1(t) - 2\dot{C}(t)\eta(t) \\ \qquad\qquad + \dot{p}_g(t)p(t) + p_g(t)\dot{p}(t) - P_2(t)\eta(t) - P_3(t) \\ \mathbf{1}^{\mathrm{T}}\dot{\eta}(t) = -\gamma Z_2(t) \end{cases} \tag{46}$$

Then setting

$$\begin{aligned} M(t) &= \begin{bmatrix} 2C(t) + P_1(t) & \mathbf{1} \\ \mathbf{1}^{\mathrm{T}} & 0 \end{bmatrix} \in \mathbb{R}^{(1+n)\times(1+n)}, \\ v(t) &= \begin{bmatrix} -\gamma Z_1(t) - 2\dot{C}(t)\eta(t) + \dot{p}_g(t)p(t) + p_g(t)\dot{p}(t) - P_2(t)\eta(t) - P_3(t) \\ -\gamma Z_2(t) \end{bmatrix} \in \mathbb{R}^{1+n}, \end{aligned} \tag{47}$$

(46) may be reformulated as below:

$$M(t)\dot{w}(t) = v(t), \tag{48}$$

where $\dot{w}(t) = \begin{bmatrix} \dot{\eta}(t) \\ \dot{\lambda}(t) \end{bmatrix} \in \mathbb{R}^{1+n}$. Since $M(t)$ is a nonsingular mass matrix, the model of (48) is adequate for addressing the MVPO1 problem of (5)–(7). However, as will be demonstrated in Section 4, since the construction of p and C in our approach in terms of MATLAB code relies on function handle, the next model of (31) is simpler to control under a MATLAB `ode` solver. As a consequence, the following is the suggested ZNN model for the MVPO1 problem of (5)–(7):

$$\dot{w}(t) = M^{-1}(t)v(t), \tag{49}$$

which, with the exception of the expense of computing the matrix's inverse, is equivalent to (48). It is important to note that using a MATLAB solver for the `ode` can effectively produce the solution $w(t)$ of Equation (49). The following theorem, Theorem 2, establishes that the ZNN solver converges to the theoretical solution.

Theorem 2. *The state vector* $w(t) = [\eta^{\mathrm{T}}(t), \lambda^{\mathrm{T}}(t)]^{\mathrm{T}}$ *of ZNN* (49) *converges universally to the theoretical solution* $w^*(t) = [\eta^{*\mathrm{T}}(t), \lambda^{*\mathrm{T}}(t)]^{\mathrm{T}}$ *starting from any initial condition* $w(0) \in \mathbb{R}^{n+k}$. *To put it another way,* $\lim_{t\to\infty}(w^*(t) - w(t)) = 0$, *while the first n components of* $x^*(t)$ *are the theoretical solution* $\eta^*(t)$ *of TVQP* (39) *and* (40).

Proof. The error matrix equation group is determined as in (43), inline with the ZNN architecture, to achieve the solution $w(t)$ of TVQP (39) and (40). Model (44) is then obtained by using the linear design formula for zeroing (43). When $t \to \infty$, each error matrix equation in the group (44) converges to zero matrix, according to ([10] Theorem 1). As a consequence, when $t \to \infty$, the solution of (44) converges to the theoretical solution of TVQP (39) and (40). Additionally, we can infer from the (49) derivation procedure that it is merely a variant of the (44) error. Thus, the proof is finished. □

3.3. ZNN Approach on the MVPO3 Problem

Similar to the ZNN approach on the MVPO1 and MVPO2 problems, the three steps listed below can be used to develop a ZNN model in order to solve the MVPO3 problem.

Step 1: **(MVPO3 problem reformulation)** The MVPO3 problem of (8)–(11) can be reformulated as follows:

$$\min_{\eta(t)} \quad -p(t)^{\mathrm{T}}\eta(t) \tag{50}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{51}$$

$$\eta^{\mathrm{T}}(t)C(t)\eta(t) \le p_g(t) \tag{52}$$

$$\eta(t) \le \mathbf{1} \tag{53}$$

$$-\eta(t) \le \mathbf{0}, \tag{54}$$

or equivalent

$$\min_{\eta(t)} \quad -p(t)^{\mathrm{T}}\eta(t) \tag{55}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1 \tag{56}$$

$$\eta^{\mathrm{T}}(t)C(t)\eta(t) \le p_g(t) \tag{57}$$

$$A(t)\eta(t) \le b(t), \tag{58}$$

where $A(t) = \begin{bmatrix} I_n \\ -I_n \end{bmatrix} \in \mathbb{R}^{2n\times n}$ and $b(t) = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n}$. Then, using the penalty function proposed in [30,31], the problem of (55)–(58) may be reformulated as below:

$$\min_{\eta(t)} \quad -p(t)^{\mathrm{T}}\eta(t) + P(\eta(t)) + P_2(\eta(t)) \tag{59}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathrm{T}}\eta(t) = 1, \tag{60}$$

where $P(\eta(t)) = h\sum_{i=1}^{2n} e^{-sN_i(t)} \in \mathbb{R}$ with $N_i(t) = b_i(t) - A_i(t)\eta(t)$, $i = 1, 2, \ldots, 2n$, and $P_2(\eta(t)) = h_2 W(t) \in \mathbb{R}$ with $W(t) = e^{-s_2(p_g(t) - \eta^{\mathrm{T}}(t)C(t)\eta(t))}$. Notice that $h, h_2 \ge 0$ and $s, s_2 > 0$, respectively, denote the penalty and the design parameter. The values of $P(\eta(t)), P_2(\eta(t))$, respectively, are more consequent with the conditions (58) as s, s_2 increase, although calculations will take longer if s, s_2 are too large. When $N(t) = 0$ and $\eta^{\mathrm{T}}(t)C(t)\eta(t) = p_g(t)$, h, h_2, respectively, can push the values of $P(\eta(t)), P_2(\eta(t))$ to be close to zero.

Step 2: **(Conditions of minimization)** The optimization problem in (59) and (60) is solved by determining the following Lagrange function:

$$L(\eta(t),\lambda(t),t) = -p(t)^{\mathrm{T}}\eta(t) + P(\eta(t)) + P_2(\eta(t)) + \lambda^{\mathrm{T}}(t)(\mathbf{1}^{\mathrm{T}}\eta(t) - 1). \tag{61}$$

The following are the first-order conditions:

$$\begin{cases} \frac{\partial L(\eta(t),\lambda(t),t)}{\partial \eta(t)} = -p(t) + P_\eta(t) + P_{2,\eta}(t) + \lambda(t)\mathbf{1} = 0 \\ \frac{\partial L(\eta(t),\lambda(t),t)}{\partial \lambda(t)} = \mathbf{1}^{\mathrm{T}}\eta(t) - 1 = 0 \end{cases} \tag{62}$$

where $P_\eta(t) = hs\sum_{i=1}^{2n}\left(e^{-sN_i(t)}A_i^{\mathrm{T}}(t)\right) \in \mathbb{R}^n$ and $P_{2,\eta}(t) = 2h_2s_2W(t)C(t)\eta(t) \in \mathbb{R}^n$.

Step 3: **(ZNN solver)** The next error matrix equation group is set:

$$\begin{cases} Z_1(t) = -p(t) + P_\eta(t) + P_{2,\eta}(t) + \lambda(t)\mathbf{1} \\ Z_2(t) = \mathbf{1}^{\mathrm{T}}\eta(t) - 1 \end{cases} \tag{63}$$

and then replacing $Z(t)$ in (12) with $Z_i(t), i = 1, 2$, defined in (63), one obtains:

$$\begin{cases} -\dot{p}(t) + \dot{P}_\eta(t) + \dot{P}_{2,\eta}(t) + \dot{\lambda}(t)\mathbf{1} + \lambda(t)\mathbf{0} = -\gamma Z_1(t) \\ \mathbf{0}^{\mathrm{T}}\eta(t) + \mathbf{1}^{\mathrm{T}}\dot{\eta}(t) = -\gamma Z_2(t) \end{cases} \tag{64}$$

where $\dot{P}_\eta(t) = P_1(t)\dot{\eta}(t) + P_2(t)\eta(t) + P_3(t)$ with $P_1(t), P_2(t), P_3(t)$ defined exactly as in (45), and $\dot{P}_{2,\eta}(t) = P_4(t)\dot{\eta}(t) + P_5(t)\eta(t) + P_6(t)$

$$\begin{aligned} P_4(t) &= 2h_2s_2W(t)(2s_2\eta^{\mathrm{T}}(t)C(t)\eta(t)C(t) + C(t)) \in \mathbb{R}^{n\times n}, \\ P_5(t) &= 2h_2s_2W(t)\dot{C}(t) \in \mathbb{R}^{n\times n}, \\ P_6(t) &= 2h_2s_2^2W(t)C(t)\eta(t)(\eta^{\mathrm{T}}(t)\dot{C}(t)\eta(t) - \dot{p}_g(t)) \in \mathbb{R}^n. \end{aligned} \tag{65}$$

It is important to mention that $\dot{A}(t) = \begin{bmatrix} O_n \\ O_n \end{bmatrix} \in \mathbb{R}^{2n\times n}$ and $\dot{b}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n}$. As a consequence, (26) may be reformulated as below:

$$\begin{cases} (P_1(t) + P_4(t))\dot{\eta}(t) + \dot{\lambda}(t)\mathbf{1} = -\gamma Z_1(t) - 2\dot{C}(t)\eta(t) \\ \qquad -(P_2(t) + P_5(t))\eta(t) - P_3(t) - P_6(t) + \dot{p}(t) \\ \mathbf{1}^{\mathrm{T}}\dot{\eta}(t) = -\gamma Z_2(t) \end{cases} \tag{66}$$

Then setting

$$\begin{aligned} M(t) &= \begin{bmatrix} P_1(t)+P_2(t) & \mathbf{1} \\ \mathbf{1}^{\mathrm{T}} & 0 \end{bmatrix} \in \mathbb{R}^{(1+n)\times(1+n)}, \\ v(t) &= \begin{bmatrix} -\gamma Z_1(t) - 2\dot{C}(t)\eta(t) - (P_2(t)+P_5(t))\eta(t) - P_3(t) - P_6(t) + \dot{p}(t) \\ -\gamma Z_2(t) \end{bmatrix} \in \mathbb{R}^{1+n}, \end{aligned} \tag{67}$$

(46) may be reformulated as below:

$$M(t)\dot{w}(t) = v(t), \tag{68}$$

where $\dot{w}(t) = \begin{bmatrix} \dot{\eta}(t) \\ \dot{\lambda}(t) \end{bmatrix} \in \mathbb{R}^{1+n}$. Since $M(t)$ is a nonsingular mass matrix, the model of (68) is adequate for addressing the MVPO1 problem of (8)–(11); however, as will be demonstrated in Section 4, since the construction of p and C in our approach in terms of MATLAB code

relies on function handle, the next model of (31) is simpler to control under a MATLAB `ode` solver. As a consequence, the following is the suggested ZNN model for the MVPO1 problem of (8)–(11):

$$\dot{w}(t) = M^{-1}(t)v(t), \tag{69}$$

which, with the exception of the expense of computing the matrix's inverse, is equivalent to (68). It is important to note that using a MATLAB solver for the `ode` can effectively produce the solution $w(t)$ of Equation (69). The following Theorem 3, establishes that the ZNN solver converges to the theoretical solution.

Theorem 3. *The state vector* $w(t) = [\eta^{\mathrm{T}}(t), \lambda^{\mathrm{T}}(t)]^{\mathrm{T}}$ *of ZNN* (69) *converges universally to the theoretical solution* $w^*(t) = [\eta^{*\mathrm{T}}(t), \lambda^{*\mathrm{T}}(t)]^{\mathrm{T}}$ *starting from any initial condition* $w(0) \in \mathbb{R}^{n+k}$. *To put it another way,* $\lim_{t\to\infty}(w^*(t) - w(t)) = 0$, *while the first n components of* $x^*(t)$ *are the theoretical solution* $\eta^*(t)$ *of the optimization problem in* (59) *and* (60).

Proof. The error matrix equation group is determined as in (63), inline with the ZNN architecture, to achieve the solution $w(t)$ of the optimization problem in (59) and (60). The model (64) is then obtained by using the linear design formula for zeroing (63). When $t \to \infty$, each error matrix equation in the group (64) converges to zero matrix, according to ([10] Theorem 1). As a consequence, when $t \to \infty$, the solution of (64) converges to the theoretical solution of the optimization problem in (59) and (60). Additionally, we can infer from the (69) derivation procedure that it is merely a variant of the (64) error. Thus, the proof is finished. □

4. Real-World Simulation Results

The time series data in the MVPO problems are the portfolio's covariance matrix and expected return array. Since the input data are in DT, they must be converted to CT. We accomplish this by employing interpolation functions. Algorithm 1 describes how we create the expected return array p and its first derivative $\dot{p}$, as well as the covariance matrix C and its first derivative $\dot{C}$. It is important to note that the input data must be translated from DT to CT since we are aiming to calculate the online solution of CT problems. In order to interpolate linearly between arrays and matrices, Algorithm 1 uses the MATLAB custom functions `linots` and `linotss`, which are retrieved from [34]. The DT arrays $p(t)$, $\dot{p}(t)$, $p_g(t)$, and $\dot{p}_g(t)$ are converted into interpolated CT functions using `linots`, and the DT matrices $C(t)$ and $\dot{C}(t)$ are converted into interpolated CT functions using `linotss`. Nevertheless, a number of additional custom interpolation functions for well-liked interpolation techniques are suggested in [34], where their major benefit over the commercial functions of MathWorks is that they are being operated more effectively by the MATLAB `ode` solvers, reducing computational expenses. To put it another way, when time series constitute the input data, the ZNN produces quicker results.

To have an appropriate covariance matrix for comparisons, it is important to note that the portfolio's data are normalized for each time period. The covariance matrix C is multiplied by 100 without loss of generality, which results in the variance of the portfolio being expressed in percentages. Additionally, the MATLAB `ode15s` solver is used to create the online solution of the MVPO1, MVPO2, and MVPO3 problems for Equations (31), (49), and (69), respectively. In addition, in the following experiments, all the parameters have been set as shown in Table 1. The financial time series utilized are retrieved from https://finance.yahoo.com (accessed on 1 February 2022) and the following link leads you to the entire implementation and development of the MVPO1, MVPO2, and MVPO3 problems discussed in Sections 3 and 4 on GitHub: https://github.com/SDMourtas/CTMVPO (accessed on 1 February 2022). Lastly, the solutions provided by the ZNN solver are contrasted to the presumptive theoretical solutions produced by the MATLAB functions `quadprog` in the MVPO1 and MVPO2 problems, and `fmincon` in the MVPO3 problem, for comparison purposes.

Table 1. The ZNN parameters for solving the CT MVPO problems.

Parameters	Case 1: 4 Stocks Portfolio		
	MVPS1	**MVPS2**	**MVPS3**
h	1×10^{-8}	1×10^{-8}	1×10^{-8}
s	1×10^{5}	3×10^{4}	3×10^{4}
h_2	-	-	2×10^{-4}
s_2	-	-	5×10^{2}
p_g	0.963	1	0.045
γ	1×10^{4}	1×10^{4}	1×10^{4}
	Case 2: 10 Stocks Portfolio		
	MVPS1	**MVPS2**	**MVPS3**
h	1×10^{-8}	1×10^{-8}	1×10^{-8}
s	6×10^{4}	3×10^{4}	3×10^{4}
h_2	-	-	2×10^{-4}
s_2	-	-	5×10^{2}
p_g	0.955	1	0.039
γ	1×10^{4}	1×10^{4}	1×10^{4}
	Case 3: 20 Stocks Portfolio		
	MVPS1	**MVPS2**	**MVPS3**
h	1×10^{-8}	1×10^{-8}	1×10^{-8}
s	6×10^{4}	3×10^{4}	3×10^{4}
h_2	-	-	1×10^{-4}
s_2	-	-	5×10^{2}
p_g	0.955	1	0.037
γ	1×10^{4}	1×10^{4}	1×10^{4}

Algorithm 1 Data preprocessing algorithm for the CT MVPO problems.

Require: The marketed space $Q = [q_1, q_2, \ldots, q_n]$, the moving average's number of time periods $\tau \leq \mu - 1$, $\tau \in \mathbb{N}$.

procedure DATA_PREP(Q, τ)
  Set $[\mu, n]$ =size(Q), p =zeros($\mu - \tau, n$), and $C\{\mu - \tau, 1\} = \{\}$
  for $i = 1 : \mu - \tau$ **do**
    Set h =max($Q(i : \tau + i - 1, :)$)
    Put $C\{i, 1\} = 100*\text{cov}(Q(i : \tau + i - 1, :)./h)$
    Put $p(i, :)$ =mean($Q(i : \tau + i - 1, :)./h$)
  end for
  for $k = 1$:length(p) $- 1$ **do**
    Set $\dot{C}\{k, 1\} = C\{k + 1, 1\} - C\{k, 1\}$ and $\dot{p}(k, :) = p(k + 1, :) - p(k, :)$
  end for
  Put $\dot{C} = @(t)\text{linotss}(\dot{C}, t)$ and $C = @(t)\text{linotss}(C, t)$
  Put $\dot{p} = @(t)\text{linots}(\dot{p}, t)'$ and $p = @(t)\text{linots}(p, t)'$
end procedure

Ensure: The CT functions C, $\dot{C}$, p and $\dot{p}$.

Three alternative portfolio setup cases are covered by the trials. In Figure 1, the portfolio cases are shown, and the marketed space includes some of the most active stocks on the US market. We consider the market $Q = [q_1, q_2, \ldots, q_s]$ in the r-th case, $r = 1, 2, 3$, where Q includes the daily prices of the s stocks shown in Figure 1 into $q_1, q_2, \ldots, q_s$, respectively, for the time period 2 April 2019 to 1 October 2019. We employ linear data interpolation in the previously mentioned time series to convert them into functions of time; we set the time delay parameter $\tau = 20$ to calculate the expected returns p and covariance C of Algorithm 1. The remaining data span the dates 1 May 2019 to 1 October 2019 and includes 107 recorded prices. May, July, and August, in particular, each had 22 recorded prices; June has 20 recorded prices, and September and October together have 21 recorded prices. To solve the omitted recorded prices problem between periods of the same division, we utilize

the parameter ω presented in [34], which splits the recorded prices to the time periods for each t inside the ZNN model. That is, we set

$$\omega = \begin{cases} 22 & ,t \in [0,1) \\ (22+20\cdot(t-1))/t & ,t \in [1,2) \\ (42+22\cdot(t-2))/t & ,t \in [2,4) \\ (86+21\cdot(t-4))/t & ,t \in [4,5] \end{cases}$$

and, then, we employ $p(\omega t), C(\omega t)$ instead of $p(t), C(t)$, and $\dot{p}(\omega t), \dot{C}(\omega t)$ instead of $\dot{p}(t), \dot{C}(t)$. As a result, we divide our time series into five monthly periods, and we set $tspan = [0,5]$ in the MATLAB `ode15s` solver to find the optimal portfolio $\eta(t)$ for the time period 1 May 2019 to 1 October 2019. That is, May to October correspond to the values 0 to 5 of the `ode15s` solver in all the figures in this section. Starting from $w(0) =$ones$(s+1,1)/s$ in (31), (49), and (69), the results are presented in Figures 2–4.

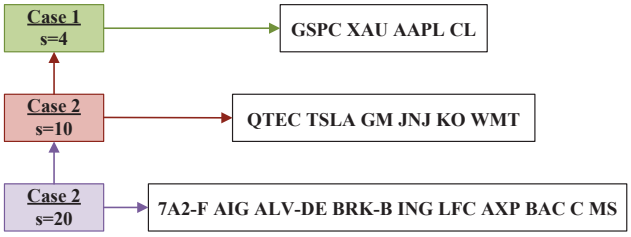
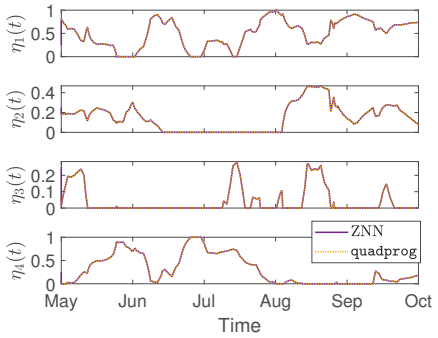
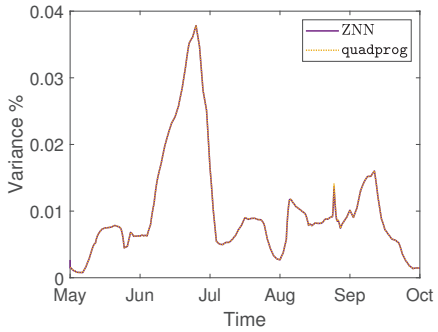


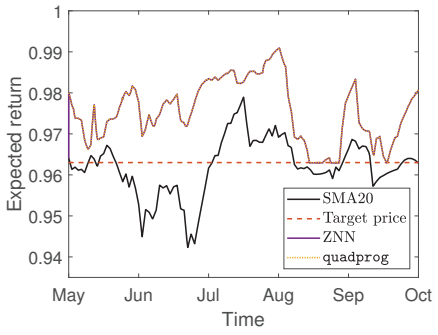
Figure 1. The portfolio cases stocks that have been utilized in the CT MVPO problems experiments.



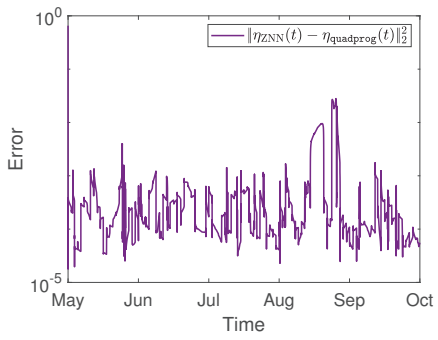
(**a**) Portfolios weights (MVPO1).



(**b**) Portfolios variance (MVPO1).

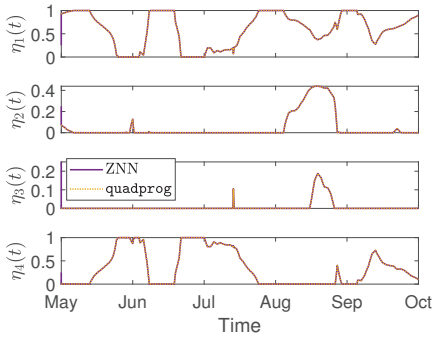


(**c**) Portfolios exp. return (MVPO1).

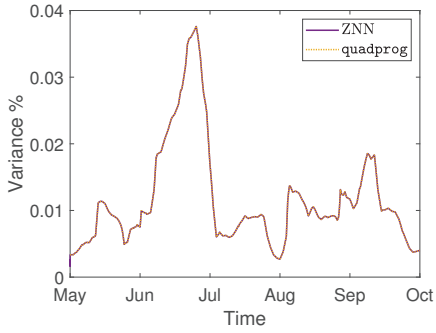


(**d**) Error: ZNN vs `quadprog` (MVPO1).

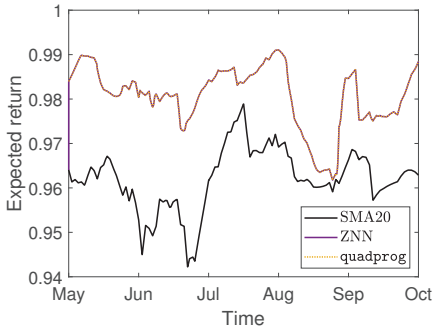
Figure 2. *Cont.*



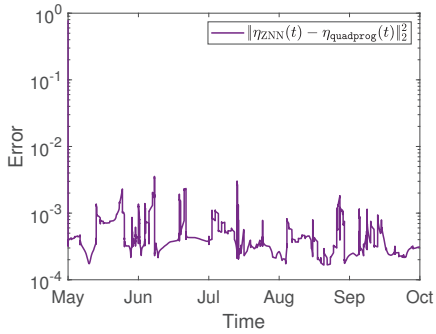
(e) Portfolios weights (MVPO2).



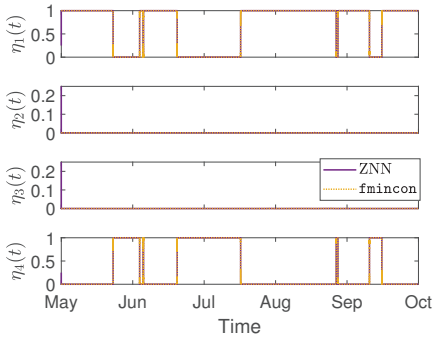
(f) Portfolios variance (MVPO2).



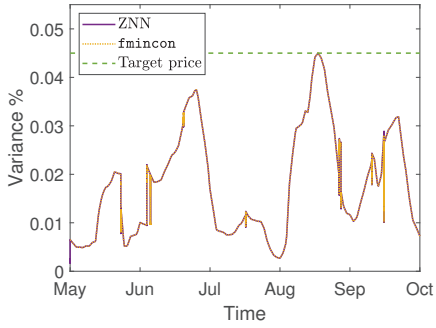
(g) Portfolios exp. return (MVPO2).



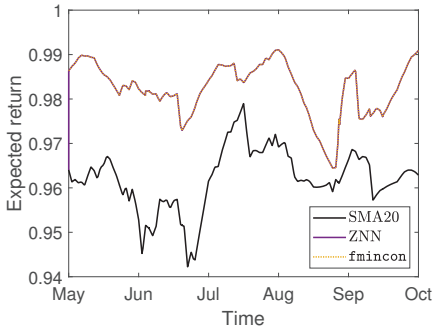
(h) Error: ZNN vs `quadprog` (MVPO2).



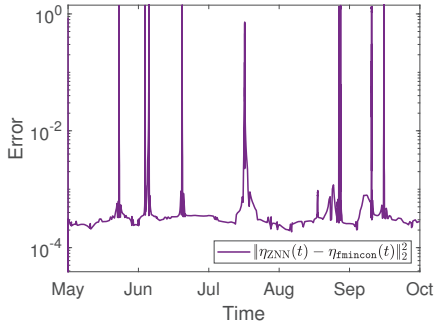
(i) Portfolios weights (MVPO3).



(j) Portfolios variance (MVPO3).

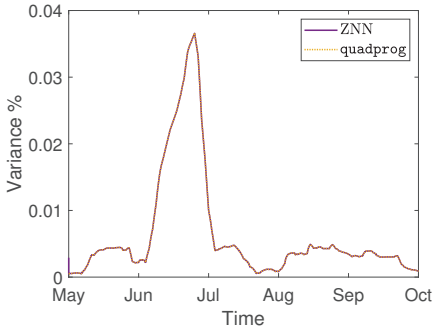


(k) Portfolios exp. return (MVPO3).

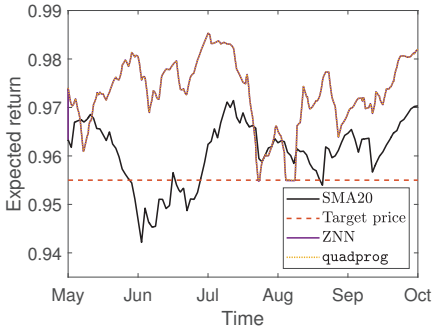


(l) Error: ZNN vs `fmincon` (MVPO3).

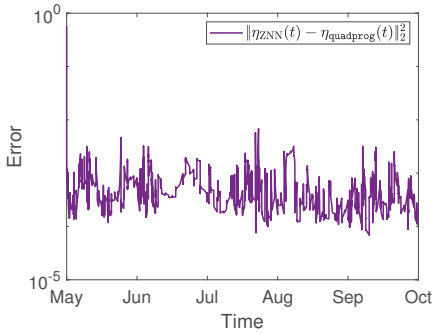
Figure 2. Portfolios weights, variance %, expected return, and the error between ZNN and MATLAB's solvers in case 1.



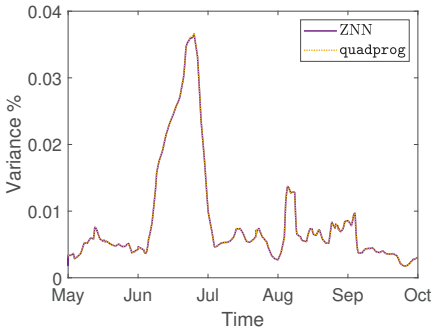
(**a**) Portfolios variance (MVPO1).



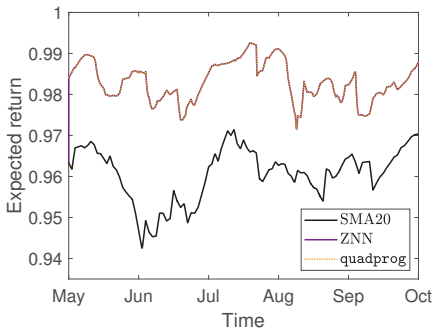
(**b**) Portfolios exp. return (MVPO1).



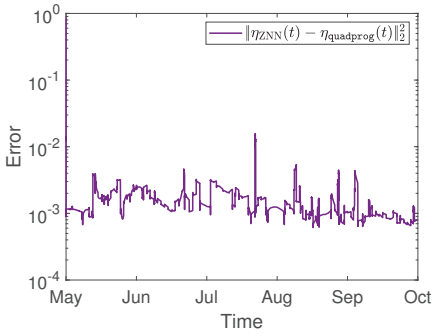
(**c**) Error: ZNN vs `quadprog` (MVPO1).



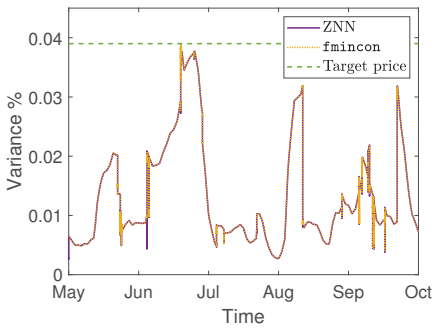
(**d**) Portfolios variance (MVPO2).



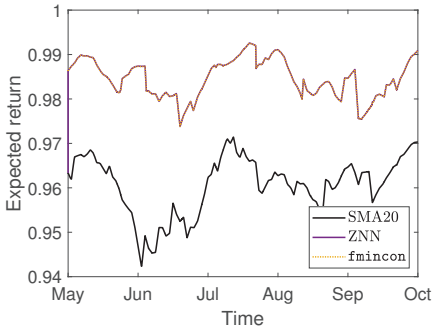
(**e**) Portfolios exp. return (MVPO2).



(**f**) Error: ZNN vs `quadprog` (MVPO2).

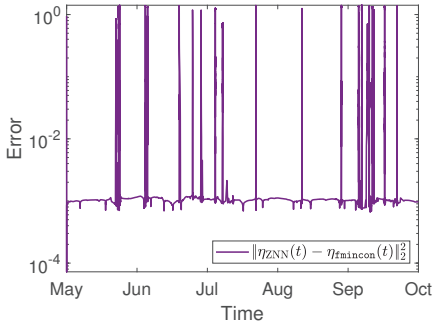


(**g**) Portfolios variance (MVPO3).



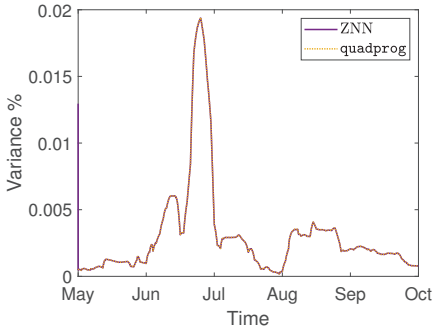
(**h**) Portfolios exp. return (MVPO3).

Figure 3. *Cont.*

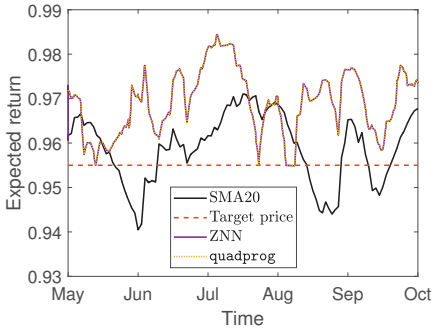


(**i**) Error: ZNN vs `fmincon` (MVPO3).

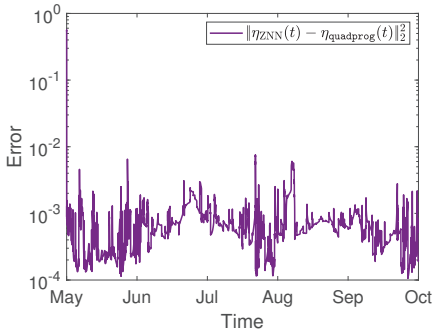
Figure 3. Portfolios variance %, expected return, and the error between ZNN and MATLAB's solvers in case 2.



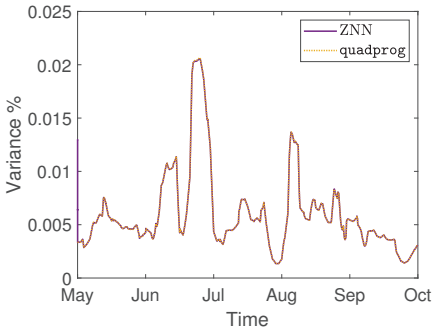
(**a**) Portfolios variance (MVPO1).



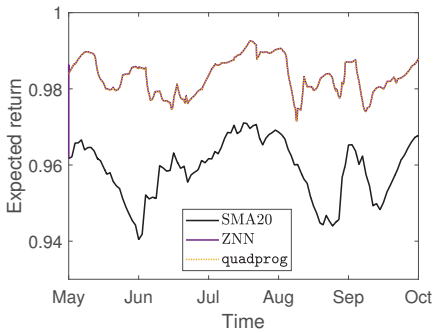
(**b**) Portfolios exp. return (MVPO1).



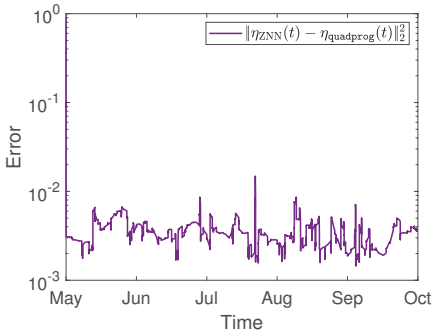
(**c**) Error: ZNN vs `quadprog` (MVPO1).



(**d**) Portfolios variance (MVPO2).

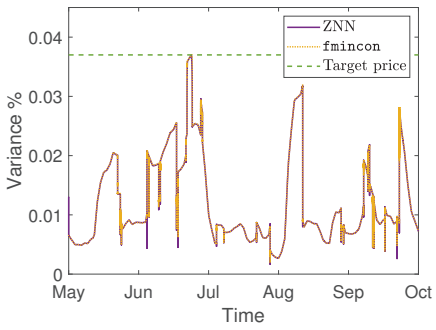


(**e**) Portfolios exp. return (MVPO2).

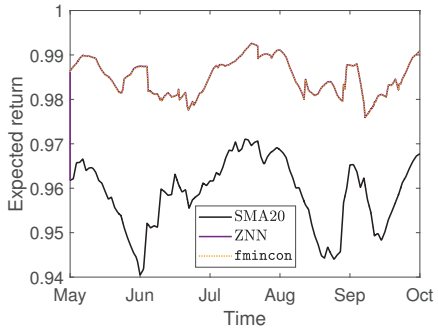


(**f**) Error: ZNN vs `quadprog` (MVPO2).

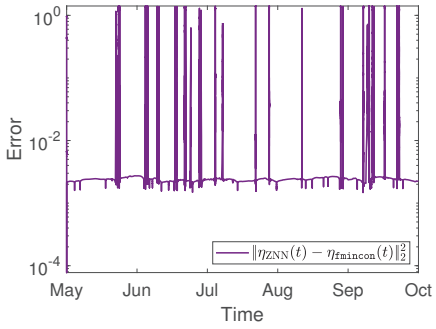
Figure 4. *Cont.*



(**g**) Portfolios variance (MVPO3).



(**h**) Portfolios exp. return (MVPO3).



(**i**) Error: ZNN vs `fmincon` (MVPO3).

Figure 4. Portfolios variance %, expected return, and the error between ZNN and MATLAB's solvers in case 3.

More precisely, Figure 2a,e,i show the optimal mean-variance portfolios weights $\eta(t)$ in the case 1 for the MVPO1, MVPO2, and MVPO3, respectively, generated by `quadprog`, `fmincon`, and ZNN. We may observe, therein, that the portfolios weights are identical. For the MVPO1, the variance of portfolios $\eta(t)$ are depicted in Figures 2b, 3a, and 4a for the cases 1, 2, and 3, respectively. The expected return of portfolios $\eta(t)$ (i.e., $\eta(t)p(\omega t)$), contrasted to the outcome of `quadprog` and the *SMA*20 of $X(t)$ (i.e., mean$(p(\omega t))$), for the cases 1, 2, and 3, respectively, are shown in Figures 2c, 3b, and 4b. These figures also show the target price $p_g(t)$. Comparing the Figures 2b, 3a, and 4a to Figures 2c, 3b, and 4b, respectively; we can observe that the variance of $\eta(t)$ is rising only in the case where it needs to keep its expected return at $p_g(t)$.

For the MVPO2, the variance of portfolios $\eta(t)$ are depicted in Figures 2f, 3d, and 4d for the cases 1, 2, and 3, respectively. The expected return of portfolios $\eta(t)$ contrasted to the outcome of `quadprog` and the *SMA*20 of $X(t)$, for the cases 1, 2, and 3, respectively, are shown in Figures 2g, 3e, and 4e. We may observe, therein, that the variance and expected return of portfolios $\eta(t)$ are identical.

For the MVPO3, the variance of portfolios $\eta(t)$, along with the target price $p_g(t)$, are depicted in Figures 2j, 3g, and 4g for the cases 1, 2, and 3, respectively. The expected return of portfolios $\eta(t)$ contrasted to the outcome of `fmincon` and the *SMA*20 of $X(t)$, for the cases 1, 2, and 3, respectively, are shown in Figures 2k, 3h, and 4h. Comparing the Figures 2j, 3g, and 4g to Figures 2k, 3h, and 4h, respectively; we can observe that the expected return of $\eta(t)$ is declining only in the case where it needs to keep its variance below the target $p_g(t)$.

Figures 2d, 3c, and 4c for the MVPO1 and Figures 2h, 3f, and 4f for the MVPO2 depict the error between ZNN and `quadprog`, produced during the ZNN convergence. In addition, Figures 2l, 3i, and 4i for the MVPO3 depict the error between ZNN and `fmincon`, produced during the ZNN convergence. It is important to note that $\eta_{\mathtt{quadprog}}(t)$, $\eta_{\mathtt{fmincon}}(t)$, and $\eta_{\mathrm{ZNN}}(t)$ in these figures represent, respectively, the portfolios produced by the `quadprog`, `fmincon`, and ZNN. Due to the non-smooth changes in portfolio weights,

there has been a sharp increase in the error depicted in Figures 2l, 3i, and 4i. Furthermore, the noise is expected since we are dealing with time series and, considering the design parameter's γ low value, the error value is excellent. Note that the overall error value of the ZNN decreases even more when the price of parameter γ increases. Our method is also more practical when taking into account the ω parameter, which comes in handy when combining time periods that each have a distinct amount of recorded prices. Overall, the portfolio cases presented in numerical experiments of this section demonstrate that the ZNN worked effectively in addressing the MVPO1, MVPO2, and MVPO3 problems.

For the purpose of monitoring the performance between the employed MATLAB custom functions (namely `linots` and `linotss`, `pchinots` and `pchinotss`, and `splinots` and `splinotss`), which are taken from [34]; we present Table 2. This table presents the average consumption time of ZNN, along with `quadprog` and `fmincon`, on each portfolio case, by using all the aforementioned MATLAB functions for linear, P.C.Hermite and C.Spline interpolation. According to Table 2, the P.C.Hermite is the least effective technique, while the linear interpolation is the most effective. Furthermore, in the MVPO1 and MVPO2 problems, the ZNN solver is always around twice as quick than the `quadprog`; however, in the MVPO3 problem, it is between 20 and 70 times quicker than the `fmincon`. It is important to mention that these MATLAB custom functions, which handle matrices and structure time series, are the best choices in terms of computing time responses, even though they yield identical results as the corresponding conventional MATLAB functions [13]. The efficacy and computational efficiency of the ZNN is proven based on this and the analyses presented in this section's numerical experiments.

Table 2. The ZNN and MATLAB's solvers time consumption for solving the MVPO problems.

Interpolation Method	Case 1: 4 Stocks Portfolio					
	MVPO1		**MVPO2**		**MVPO3**	
	ZNN	`quadprog`	ZNN	`quadprog`	ZNN	`fmincon`
Linear	0.6 s	1.4 s	0.6 s	1.1 s	0.7 s	35 s
P.C.Hermite	1 s	1.3 s	0.4 s	1 s	1 s	23 s
C.Spline	0.4 s	1.1 s	0.4 s	0.9 s	0.6 s	19 s
	Case 2: 10 Stocks Portfolio					
	ZNN	`quadprog`	ZNN	`quadprog`	ZNN	`fmincon`
Linear	1.3 s	3 s	1 s	2.6 s	2 s	144 s
P.C.Hermite	2.1 s	2.7 s	1.8 s	2.8 s	3.8 s	140 s
C.Spline	1.4 s	2.6 s	1.4 s	2.3 s	2.9 s	135 s
	Case 3: 20 Stocks Portfolio					
	MVPO1		**MVPO2**		**MVPO3**	
	ZNN	`quadprog`	ZNN	`quadprog`	ZNN	`fmincon`
Linear	2.8 s	4.3 s	2 s	3.8 s	5 s	355 s
P.C.Hermite	4.9 s	5 s	4.4 s	4.6 s	9 s	300 s
C.Spline	2.7 s	4.3 s	3.4 s	4.5 s	5.7 s	290 s

5. Conclusions

Three time-varying MVPO problems are introduced in this paper, as well as three novel ZNN models for addressing them. The focus of this research was on the use of NN computational techniques to address the MVPO1, MVPO2, and MVPO3 problems accurately in a short amount of time. Simulations with real-world data have proved the efficiency of the NN solver in financial TVQP and TVNLP problems. According to the results of the simulations, which included three various portfolio configuration cases, the ZNN models did exceptionally well in addressing the CT MVPO problems, demonstrating their utility in practical circumstances. A study limitation was the maximum portfolio dimension because the ZNN approach is unable to handle problems with very big dimensions.

There are a few prospective study areas that can be explored.

1. The use of NN solvers in higher-dimensional portfolios and in a variety of financial portfolio optimization tasks.

2. The ZNN solver's performance in real-world data problems utilizing varied activation functions.
3. Due of the importance to real financial markets, future research should concentrate on problems with more realistic and practical constraints.

Author Contributions: S.D.M.: conceptualization, methodology, validation, formal analysis, investigation, and writing—original draft. C.K.: methodology, formal analysis, and investigation. All authors have read and agreed to the published version of the manuscript.

Funding: The publication of this article has been financed by the Research Committee of the University of Patras.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Matsumoto, K. Option Replication in Discrete Time with Illiquidity. *Appl. Math. Financ.* **2013**, *20*, 167–190. [CrossRef]
2. Yu, J.R.; Chiou, W.J.P.; Lee, W.Y.; Lin, S.J. Portfolio models with return forecasting and transaction costs. *Int. Rev. Econ. Financ.* **2020**, *66*, 118–130. [CrossRef]
3. Holden, H.; Holden, L. Optimal rebalancing of portfolios with transaction costs. *Stochastics Int. J. Probab. Stoch. Process.* **2012**, *85*, 371–394. [CrossRef]
4. Annaert, J.; Ceuster, M.D.; Vandenbroucke, J. Mind the Floor: Enhance Portfolio Insurance without Borrowing. *J. Investig.* **2019**, *28*, 39–50. [CrossRef]
5. Matsumoto, K. Portfolio Insurance with Liquidity Risk. *Asia-Pac. Financ. Mark.* **2008**, *14*, 363. [CrossRef]
6. Mourtas, S.D.; Katsikis, V.N. Exploiting the Black-Litterman framework through error-correction neural networks. *Neurocomputing* **2022**, *498*, 43–58. [CrossRef]
7. Leung, M.F.; Wang, J. Minimax and Biobjective Portfolio Selection Based on Collaborative Neurodynamic Optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2825–2836. [CrossRef]
8. Yaman, I.; Dalkılıç, T.E. A hybrid approach to cardinality constraint portfolio selection problem based on nonlinear neural network and genetic algorithm. *Expert Syst. Appl.* **2021**, *169*, 114517. [CrossRef]
9. Wang, H.; Zhou, X.Y. Continuous-time mean-variance portfolio selection: A reinforcement learning framework. *Math. Financ.* **2020**, *30*, 1273–1308. [CrossRef]
10. Zhang, Y.; Ge, S.S. Design and analysis of a general recurrent neural network model for time-varying matrix inversion. *IEEE Trans. Neural Netw.* **2005**, *16*, 1477–1490. [CrossRef]
11. Kornilova, M.; Kovalnogov, V.; Fedorov, R.; Zamaleev, M.; Katsikis, V.N.; Mourtas, S.D.; Simos, T.E. Zeroing neural network for pseudoinversion of an arbitrary time-varying matrix based on singular value decomposition. *Mathematics* **2022**, *10*, 1208. [CrossRef]
12. Jiang, W.; Lin, C.L.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Simos, T.E. Zeroing neural network approaches based on direct and indirect methods for solving the Yang–Baxter-like matrix equation. *Mathematics* **2022**, *10*, 1950. [CrossRef]
13. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying mean-variance portfolio selection problem solving via LVI-PDNN. *Comput. Oper. Res.* **2022**, *138*, 105582. [CrossRef]
14. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Zhang, Y. Solving complex-valued time-varying linear matrix equations via QR decomposition with applications to robotic motion tracking and on angle-of-arrival localization. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 3415–3424. [CrossRef]
15. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Zhang, Y. Continuous-time varying complex QR decomposition via zeroing neural dynamics. *Neural Process. Lett.* **2021**, *53*, 3573–3590. [CrossRef]
16. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S. Unique non-negative definite solution of the time-varying algebraic Riccati equations with applications to stabilization of LTV systems. *Math. Comput. Simul.* **2022**, *202*, 164–180. [CrossRef]
17. Canakgoz, N.A.; Beasley, J.E. Mixed-integer programming approaches for index tracking and enhanced indexation. *Eur. J. Oper. Res.* **2009**, *196*, 384–399. [CrossRef]
18. Branke, J.; Scheckenbach, B.; Stein, M.; Deb, K.; Schmeck, H. Portfolio optimization with an envelope-based multiobjective evolutionary algorithm. *Eur. J. Oper. Res.* **2009**, *199*, 684–693. [CrossRef]
19. Nobre, J.; Neves, R.F. Combining principal component analysis, discretewavelet transform and xgboost to trade in the financial markets. *Expert Syst. Appl.* **2019**, *125*, 181–194. [CrossRef]
20. Akbay, M.A.; Kalayci, C.B.; Polat, O. A parallel variable neighborhood search algorithm with quadratic programming for cardinality constrained portfolio optimization. *Knowl.-Based Syst* **2020**, *198*, 105944. [CrossRef]

21. Uhlig, F.; Zhang, Y. Time-varying matrix eigenanalyses via Zhang neural networks and look-ahead finite difference equations. *Linear Algebra Its Appl.* **2019**, *580*, 417–435. [CrossRef]
22. Markowitz, H. Portfolio selection. *J. Financ.* **1952**, *7*, 77–91.
23. Dai, Z. A Closer Look at the Minimum-Variance Portfolio Optimization Model. *Math. Probl. Eng.* **2019**, *2019*, 1–8. [CrossRef]
24. Cornuejols, G.; Tütüncü, R. *Optimization Methods in Finance*; Cambridge University Press: Cambridge, UK, 2006. [CrossRef]
25. Bielecki, T.R.; Jin, H.; Pliska, S.R.; Zhou, X.Y. Continuous-time mean-variance portfolio selection with bankruptcy prohibition. *Math. Financ.* **2005**, *15*, 213–244. [CrossRef]
26. Draviam, T.; Chellathurai, T. Generalized Markowitz mean-variance principles for multi-period portfolio-selection problems. *Proc. R. Soc. Lond. A* **2002**, *458*, 2571–2607. [CrossRef]
27. Zakamulin, V. *Market Timing with Moving Averages: The Anatomy and Performance of Trading Rules*; Springer: Berlin/Heidelberg, Germany, 2017.
28. Markowitz, H.M. *Portfolio Selection: Efficient Diversification of Investments*; Cowles Foundation Monograph: No. 16; Yale University Press: New Haven, CT, USA, 1959; p. 368.
29. Zhang, Y.; Wang, Y.; Chen, D.; Peng, C.; Xie, Q. Neurodynamic solvers robotic applications and solution nonuniqueness of linear programming. In *Linear Programming: Theory, Algorithms and Applicant*; Nova Science Publishers:Hauppauge, NY, USA, 2014; pp. 27–100.
30. Zhong, N.; Huang, Q.; Yang, S.; Ouyang, F.; Zhang, Z. A Varying-Parameter Recurrent Neural Network Combined With Penalty Function for Solving Constrained Multi-Criteria Optimization Scheme for Redundant Robot Manipulators. *IEEE Access* **2021**, *9*, 50810–50818. [CrossRef]
31. Zhang, Z.; Yang, S.; Zheng, L. A Penalty Strategy Combined Varying-Parameter Recurrent Neural Network for Solving Time-Varying Multi-Type Constrained Quadratic Programming Problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2993–3004. [CrossRef]
32. Jin, L.; Zhang, Y.; Li, S.; Zhang, Y. Modified ZNN for Time-Varying Quadratic Programming With Inherent Tolerance to Noises and Its Application to Kinematic Redundancy Resolution of Robot Manipulators. *IEEE Trans. Ind. Electron.* **2016**, *63*, 6978–6988. [CrossRef]
33. Zhang, Y.; Wang, J. Recurrent neural networks for nonlinear output regulation. *Automatica* **2001**, *37*, 1161–1173. [CrossRef]
34. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying mean-variance portfolio selection under transaction costs and cardinality constraint problem via beetle antennae search algorithm (BAS). *SN Oper. Res. Forum* **2021**, *2*, 18. [CrossRef]

mathematics

MDPI

Article

Adopting Nonlinear Activated Beetle Antennae Search Algorithm for Fraud Detection of Public Trading Companies: A Computational Finance Approach

Bolin Liao [1], Zhendai Huang [1], Xinwei Cao [2] and Jianfeng Li [1,*]

[1] College of Information Science and Engineering, Jishou University, Jishou 416000, China; bolinliao@jsu.edu.cn (B.L.); hzd@stu.jsu.edu.cn (Z.H.)
[2] School of Management, Shanghai University, Shanghai 200000, China; xinweicao@shu.edu.cn
* Correspondence: ljf_zy@163.com; Tel.: +86-135-7431-8900

Abstract: With the emergence of various online trading technologies, fraudulent cases begin to occur frequently. The problem of fraud in public trading companies is a hot topic in financial field. This paper proposes a fraud detection model for public trading companies using datasets collected from SEC's Accounting and Auditing Enforcement Releases (AAERs). At the same time, this computational finance model is solved with a nonlinear activated Beetle Antennae Search (NABAS) algorithm, which is a variant of the meta-heuristic optimization algorithm named Beetle Antennae Search (BAS) algorithm. Firstly, the fraud detection model is transformed into an optimization problem of minimizing loss function and using the NABAS algorithm to find the optimal solution. NABAS has only one search particle and explores the space under a given gradient estimation until it is less than an "Activated Threshold" and the algorithm is efficient in computation. Then, the random under-sampling with AdaBoost (RUSBoost) algorithm is employed to comprehensively evaluate the performance of NABAS. In addition, to reflect the superiority of NABAS in the fraud detection problem, it is compared with some popular methods in recent years, such as the logistic regression model and Support Vector Machine with Financial Kernel (SVM-FK) algorithm. Finally, the experimental results show that the NABAS algorithm has higher accuracy and efficiency than other methods in the fraud detection of public datasets.

Keywords: fraud detection; nonlinear activated beetle antennae search; unbalanced dataset; computational finance

MSC: 91G40; 90C26

1. Introduction

In recent years, people rely more on the Internet for the development of modern technologies, which not only brings great convenience to people's lives but also brings great risks. In the financial field, it has led to a lot of fraudulent financial activities [1], such as mobile payment fraud [2], telecom fraud [3] and credit card fraud [4]. The detection of fraudulent financial activities is thus necessary. In [5], Ngai et al. proposed that financial fraud detection (FFD) distinguishes false financial information from real data and reveals financial fraud activities, which plays an important role in preventing the serious consequences of financial fraud. In particular, the fraud detection [6] of public trading companies is a hot issue in FDD. In [7], Reurink pointed out that company managers may provide false financial statements due to insider trading, performance pressure, or financial difficulties. The fraudulent behavior will not only directly harm the interests of investors, but also affect fair competition with other non-fraudulent companies. However, due to the wide gap between fraudulent companies and non-fraudulent companies, and the complexity

and diversity of fraudulent activities, it is difficult to judge fraudulent companies in a short time.

In order to achieve the correct classification of the specified type, huge datasets are used for training. In general, the datasets used for classification are divided into the labeled and unlabeled dataset. Machine learning (ML) is a method that learns from training datasets to complete prediction tasks. It is an effective solution for fraud detection and prevention, and is now increasingly used in the fraud detection of the financial industry. Based on unlabeled datasets, an unsupervised machine learning method named autoencoder was used to accurately detect fraudulent events [8]. Meanwhile, other ML methods include decision tree, random forest, and XGBoost algorithm, which based on labeled datasets were also used in fraud detection [9]. According to the efficient classification and prediction of machine learning, in this paper, a model based on ML methods is proposed, and the confusion matrix in machine learning and its related evaluation indicators are used to evaluate the performance of the model.

It is worth noting because of the difficulty for technicians to gather enough useful information from the real environment, learning and analysis based on unbalanced datasets are ubiquitous [10]. Meanwhile, in [11], various ML methods were used for the classification and prediction of unbalanced datasets. The results showed that no matter which method is used, the impact of unbalanced datasets on prediction accuracy cannot be eliminated. Therefore, it is necessary to formulate specific strategies for special unbalanced datasets. This paper employs an over-sampling technique to deal with unbalanced datasets and sets an appropriate threshold to adjust for the bias between fraudulent and non-fraudulent companies. In this way, the number of samples of fraudulent and non-fraudulent companies can be balanced as much as possible, thereby reducing the impact of unbalanced datasets on the classification and prediction results of ML methods.

Different from the existing ML methods mentioned above, this paper presents an improved Beetle Antennae Search (BAS) algorithm to solve the optimization problem. The Beetle Antennae Search algorithm is a meta-heuristic algorithm, which was proposed by Jiang et al. [12]. It simulates the behavior of the beetles using antennae to search for food in nature and only has one particle for global search. Thus, this algorithm has the advantage of quick convergence properties and simple implementation compared to other swarm optimization algorithms [13]. In recent years, in the financial field, the BAS algorithm has been applied to the portfolio insurance problem [14–19]. However, the research on public trade audit companies is limited and few studies use the BAS algorithm for fraud detection [20]. Additionally, the original BAS algorithm has shortcomings like premature convergence and easy falling into local minima. In this paper, a variant of BAS named nonlinear activated Beetle Antennae Search (NABAS) is presented to solve the fraud detection problem of public trading companies. The NABAS algorithm introduces a nonlinear "Activated Threshold" based on the step size update rule of BAS. When the gradient value is less than the threshold, the algorithm is allowed to update the search step size. This improvement helps NABAS avoid a fast convergence rate and be immune to local minima.

Over the years, many studies have proposed solutions to the problem of fraud detection in public trading companies. Dechow et al. created a dataset from 1982 to 2005 about fraudulent companies, which was collected from Accounting and Auditing Enforcement Releases (AAERs) issued by the Securities and Exchange Commission (SEC) [21]. The dataset summarized the characteristics that could best distinguish fraudulent companies from non-fraudulent companies, and these characteristics were used as variables in the logistic regression model based on financial ratios. After that, a financial kernel was created by Cecchini in [22], which could implicitly map relevant financial attributes to ratios. Then, the constructed kernel function was used as part of a support vector machine (SVM-FK) to detect fraud. Moreover, a new ensemble learning method named Random Under-Sampling with AdaBoost (RUSBoost) was adopted in [23], and it is a boosting-based sampling algorithm for dealing with unbalanced data in class tag datasets. According to the

fraud detection in public trading companies in this paper, the three methods including the logistic regression model, SVM-FK, and RUSBoost are used as the benchmark to compare with NABAS.

The rest of this paper is organized as follows: Section 2 introduces the efficient way to handle an unbalanced dataset and formulate a model for this fraud detection problem. Section 3 introduces the original BAS algorithm and further presents the NABAS algorithm. Section 4 applies the NABAS algorithm to the fraud detection model and compared it with RUSBoost, logistic regression model, and SVM-FK. Section 5 summarizes the work of this paper.

2. Unbalanced Dataset and Model Establishment

In this section, firstly, the nature and processing method of the highly unbalanced dataset are given. Moreover, combined with the confusion matrix, the performance evaluation indicators of the model are analyzed. Finally, a loss function is designed to get the optimal solution to this fraud detection problem.

2.1. Highly Unbalanced Dataset

In this paper, we use a typical unbalanced dataset, which is collected from SEC's AAERs [23]. It contains a total of 206,026 public trading USA companies, with 204,855 non-fraudulent companies but only 1171 fraudulent companies from 1979 to 2014. The dataset is shown in Table 1. Particularly, the dataset contains 28 raw financial variables and 14 financial ratios.

Table 1. Distribution of fraudulent companies by year from 1979–2014 [23].

Year	Total Companies	Fraudulent Companies	Percentage of Fraudulent Companies (%)
1979	3782	4	0.11
1980	4010	10	0.25
1981	4501	12	0.27
1982	4718	19	0.40
1983	5056	14	0.28
1984	5100	16	0.31
1985	5087	10	0.20
1986	5234	21	0.40
1987	5406	16	0.30
1988	5129	19	0.37
1989	4977	23	0.46
1990	4893	18	0.37
1991	4981	28	0.56
1992	5209	28	0.54
1993	5644	31	0.55
1994	5966	24	0.40
1995	6561	22	0.34
1996	7095	34	0.48
1997	7138	45	0.63
1998	7042	56	0.80
1999	7230	79	1.09
2000	7153	92	1.29
2001	6780	87	1.28
2002	6475	81	1.25
2003	6285	69	1.10
2004	6218	60	0.96
2005	6119	47	0.77
2006	6130	35	0.57
2007	6073	30	0.49
2008	5817	27	0.46

Table 1. *Cont.*

Year	Total Companies	Fraudulent Companies	Percentage of Fraudulent Companies (%)
2009	5618	30	0.53
2010	5585	26	0.47
2011	5583	22	0.39
2012	5814	21	0.36
2013	5831	11	0.19
2014	5786	4	0.07
Total	206,026	1171	0.57

In [23], the financial ratios applied to the fraud prediction model are more powerful than the raw financial variables. However, the ratios were identified by experts; it thus lacks lots of useful information. In order to ensure the integrity of the dataset, these 28 raw financial variables are used as the dataset for the fraud detection of public trading companies. This means we have to handle a 28-dimensional dataset which is complex to show each component of it. To display the data information more clearly, a technique called principal component analysis (PCA) is used to visualize this dataset [24]. The characteristic of PCA is to extract the main feature components of data from complex high-dimensional information, which contains most of the useful information in the dataset. After processing by PCA, two of the principal components were extracted from the 28-dimensional dataset, and the results are displayed in Figure 1a. From the picture, one can find that the information among the samples overlaps, and the number of two categories has a huge gap; it is difficult to deal with them directly. Thus, formulating a targeted sample processing strategy for the specific issue is necessary.

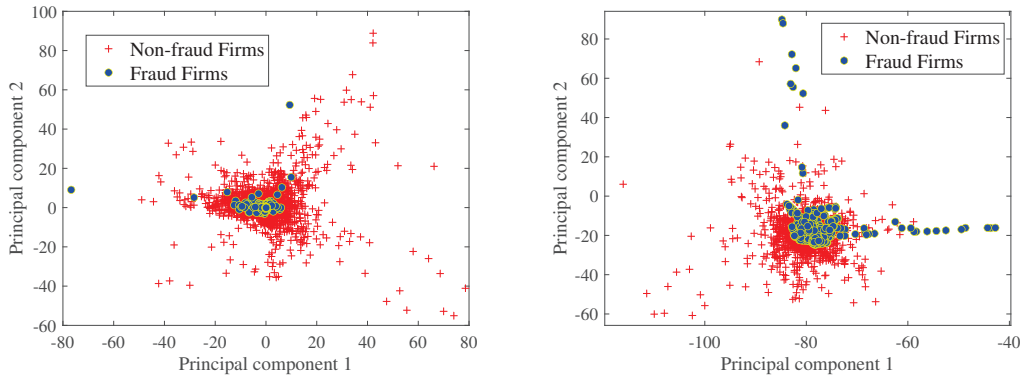


Figure 1. Visualization of the unbalanced dataset, the red plus signs represent non-fraudulent companies and the blue points represent fraudulent companies. (**a**) Visualization of 28-dimensional samples in the original dataset; (**b**) visualization of the dataset after the samples are processed by a SMOTE technique.

Here are several sampling techniques of processing to deal with the unbalanced dataset, e.g., under-sampling techniques, over-sampling techniques, and another over-sampling technique named Synthetic Minority Over-sampling Technique (SMOTE) [25]. The simple introductions of these techniques are as follows:

2.1.1. Under-Sampling Technique

For majority classes in unbalanced datasets, the under-sampling technique randomly reduces the number of majority classes until it was consistent with the minority class. In this dataset, we should use the under-sampling technique to reduce 204,855 non-fraudulent

companies to 1171, which is equivalent to fraudulent companies. This technique is easy to implement, but a large amount of useful data will be lost in the under-sampling process.

2.1.2. Over-Sampling Technique

The process of this technique is oppositive to under-sampling, which aims at increasing the number of the minority classes, by directly repeating the relevant information until the number is equal to the majority classes. This is an efficient way to balance the numbers of these two categories. However, after over-sampling dealing with the unbalanced dataset, the class obtained from the minority class is highly repetitive, and the prediction result is easy to over-fit.

2.1.3. Over-Sample Data with SMOTE Technique

At present, the over-sampling technique is used frequently in various fields. However, blindly adding new samples may be ineffective or produce noise to the dataset. In this problem, the SMOTE technique is used to oversample the minority class. Firstly, for each minority class sample, SMOTE takes the Euclidean distance as the standard and selects another sample from its nearest neighbor. In addition, it then randomly generates a point on the connection line between the two as the new composite sample. Finally, iterating continuously, until the few classes are sampled to most classes. Compared with the conventional over-sampling technique, SMOTE helps the classification models obtain higher prediction accuracy. It combines the characteristics of different samples to provide a diversity of data better. After tackling with SMOTE, the new dataset is shown in Figure 1b. It can be seen that the number of fraud companies has increased and the overlap between new samples has decreased.

2.2. Model Evaluation

In machine learning, it is crucial to choose an evaluation method that matches the problem, which helps to quickly find the key in the model selection or training process. For the binary confusion matrix, true positive (TP) is the number of fraudulent companies that are correctly classified as fraudulent companies, true negative (TN) is the number of non-fraudulent companies that are classified as non-fraudulent companies, false positive (FP) is the number of non-fraudulent companies that are misclassified as fraudulent companies, and false negative (FN) is the number of fraudulent companies that are misclassified as non-fraudulent companies [26]. This paper chooses four evaluation indicators that are closely related to the binary confusion matrix to evaluate the performance of the model. (1) Accuracy: Accuracy refers to the proportion of correctly classified samples to the total number of samples. It is the most basic index of all model evaluation indicators. However, when the sample proportion in different categories is extremely unbalanced, the category with a large proportion tends to be the most important factor affecting the accuracy. (2) Precision and Sensitivity: Precision refers to the proportion of the number of positive samples that are correctly classified to the number of positive samples determined by the classifier. Sensitivity refers to the ratio of the number of positive samples which are correctly predicted to the number of samples labeled as positive. Precision requires the detection of positive samples as accurately as possible, and sensitivity is expected to test the positive samples more comprehensively. The higher the value of both, the better the performance of the model. Precision and sensitivity are negatively correlated, so we need to find a balance to make them as high as possible at the same time. (3) NDCG: NDCG is a normalization of the Discounted Cumulative Gain (DCG) measure; it is usually used to measure and evaluate search results [27]. The DCG at the position k is defined as $\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{r(i)}{\log_2(i+1)}$, where $r(i)$ represents the correlation score of each item in the list and k is the number of top item in the list. Then, the NDCG at the position k is calculated by $\mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{rank\ DCG@}k}$, where rank DCG@$k$ is the DCG value when all the correctly predicted positive samples are ranked at the top of the ranking list.

2.3. Optimization Problem

Since this problem is only necessary to judge whether the type of a public trading company is a fraudulent company, it is a labeled binary classification problem.

2.3.1. Binary Classification

In a binary classification problem of dimension n, the feature sample $\mathbf{p} \in \mathbb{R}^{n\times 1}$ and its corresponding label $\mathbf{y} \in \mathbb{R}^{n\times 1}$. In general, a linear decision function is used to tackle the binary classification problem, which is presented as

$$f(\mathbf{p}) = \mathbf{a}^{\mathrm{T}}\mathbf{p} + b, \tag{1}$$

where $\mathbf{a} \in \mathbb{R}^{n\times 1}$ is the decision variables and b is a bias. Substituting (1) into the sigmoid function $g(p) = 1/(1+e^{-p})$, we obtain

$$g_{\mathbf{a}}(\mathbf{p}) = g\left(\mathbf{a}^{\mathrm{T}}\mathbf{p} + b\right) = \frac{1}{1+e^{-\left(\mathbf{a}^{\mathrm{T}}\mathbf{p}+b\right)}},$$

which maps the decision function (1) to $(0,1)$.

Because of the special characteristics of the unbalanced dataset, even though the dataset is handled by SMOTE, there is still a great difference between the number of fraudulent companies and non-fraudulent companies. Therefore, a threshold $\beta \in (0,1)$ is introduced to help classify the two categories. Here, we can obtain a new decision rule as

$$h_{\mathbf{a}}(\mathbf{p}) = \begin{cases} 1, g_{\mathbf{a}}(\mathbf{p}) \geq \beta \\ 0, g_{\mathbf{a}}(\mathbf{p}) < \beta \end{cases},$$

where $h_{\mathbf{a}}(\mathbf{p}) = 0$ expresses non-fraudulent companies and $h_{\mathbf{a}}(\mathbf{p}) = 1$ expresses fraudulent companies. Generally, when $\beta = 0.5$, the probability of judging two categories is equal. However, in this dataset, it is known that there exist characteristics like nonlinear and highly overlapping between each variable. To make the whole decision rule more inclined towards the category determination of fraudulent companies, β will be set smaller than 0.5. The impact of the unbalanced dataset can thus be reduced.

2.3.2. Loss Function

In general, for classification and regression problems, the loss function is defined as

$$F(\mathbf{y}, h) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\mathbf{a}}(p_i) - y_i\right)^2,$$

which is used to minimize to get the optimal value for each parameter of the model. In this problem, the loss function on the results of two different classes needs to be calculated at the same time. Then, we improve the function to a convex function, to help obtain the global optimal solution. The loss function is finally determined as

$$F(\mathbf{y}, h) = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log h_{\mathbf{a}}(p_i) + (1-y_i)\log(1-\log h_{\mathbf{a}}(p_i))\right]. \tag{2}$$

In addition, we consider true positive rate (TPR) and false positive rate (FPR) as

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},$$
$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}.$$

For this problem, TPR shows that the model correctly predicts the number of fraudulent companies in the dataset, while FPR is the opposite of TPR.

Then, the accuracy $\overline{\text{ACC}}$ is used for this model; it is aimed at correctly predicting the number of positive class (fraud) and negative class (non-fraud) as much as possible at the same time, which is defined as

$$\overline{\text{ACC}} = \frac{\text{TPR} + (1 - \text{FPR})}{2}.$$

The higher the accuracy, the better the model performance. While the loss function needs to obtain the minimization of the solution, we should make $\overline{\text{ACC}}$ to be reciprocal, and it is finally given as

$$\Gamma\left(\overline{\text{ACC}}\right) = \frac{2}{\text{TPR} + (1 - \text{FPR})}, \tag{3}$$

where $\Gamma(\cdot)$ represents the reciprocal of $\overline{\text{ACC}}$. Giving different weights λ_i to formulas (2) and (3), and adding them together, one can obtain a new loss function:

$$W\left(\mathbf{y}, h, \overline{\text{ACC}}\right) = \lambda_1 F(\mathbf{y}, h) + \lambda_2 \Gamma\left(\overline{\text{ACC}}\right). \tag{4}$$

The loss function (4) is transformed into an optimization problem with a constraint

$$\begin{aligned} & \min_{\theta} W\left(\mathbf{y}, h, \overline{\text{ACC}}\right), \\ \text{Subject to :} \quad & \\ & 0 \leq a_i \leq 1, i = 1, 2, \ldots, n. \end{aligned} \tag{5}$$

where the a_i is the ith element of vector $\mathbf{a}$. Finally, the fraud detection problem of public trading companies is converted to the above optimization problem.

Remark 1. *In machine learning, some algorithms such as back-propagation need to calculate the derivative of the loss function to update the weights and then obtain the minimum error. Different from back-propagation, the meta-heuristic algorithms like PSO, BAS and ABC et al. have the advantage of obtaining the optimal or feasible solution to a problem based on intuition or experience. The meta-heuristic algorithms are effective for a function whether it is differentiable or not. They only need to compare the loss value obtained by each iteration with the minimum value. Finally, they can reach convergence and obtain the global minimum.*

3. Nonlinear Activated Beetle Antennae Search (NABAS)

In this section, we will introduce the original BAS algorithm and further give the NABAS algorithm.

3.1. BAS

In the nature, beetles use two antennaes on their head (i.e., the right antennae and left antennae) to receive scents. By comparing the concentration of scent received by the two antennaes, the beetle determines the direction of the next movement, and finally finds the location of the food. The searching behaviors of both right-hand and left-hand are presented as

$$\mathbf{p}_r = \mathbf{p}_m + \zeta_m \times \mathbf{u}, \mathbf{p}_l = \mathbf{p}_m - \zeta_m \times \mathbf{u},$$

where $\mathbf{p}_m$ denotes the position of the particle, $\mathbf{p}_r$ denotes a position lying in the searching area of the right-hand side, and $\mathbf{p}_l$ denotes a position lying in the searching area of the left-hand side, ζ_m is the sensing length of antennae, and $\mathbf{u}$ is a random directional vector.

In the optimization problem (5), the position of the particle $\mathbf{p}$ is denoted to be $W(\mathbf{p})$, i.e., the loss function (4). After getting the scent concentration of $\mathbf{p}_r$ and $\mathbf{p}_l$, the position of the particle needed to change. The updated criteria of BAS depends on gradient estimate measure, which is presented as

$$\Delta W = W(\mathbf{p}_r) - W(\mathbf{p}_l),$$

where ΔW is the difference between the two sides of the scents, which determines the next step of the particle. The next position of the particle is

$$\mathbf{p}_{m+1} = \mathbf{p}_m \pm \delta_m \times \text{sign}[\Delta W] \times \mathbf{u}, \tag{6}$$

where sign$(\cdot)$ represents a sign function, and δ_m is the step size of searching, which accounts for the speed of particle convergence. After each iteration, ζ_m and δ_m need to be updated, so that the particle can reach the source of the scent at the final. The symbol takes "$+$" when the algorithm is used for maximization problems and uses "$-$" for minimization problems. Because the optimization problem of this paper is finding the minimum value, here we use "$-$" in (6). Lastly, the updated rules of ζ_m and δ_m are as follows:

$$\zeta_{m+1} = 0.95\zeta_m + 0.01, \delta_{m+1} = 0.95\delta_m.$$

3.2. NABAS

BAS is a non-deterministic algorithm with the random factor **u**; this factor helps BAS avoid falling into the local minimum in a certain direction, and thus obtains the ability of global searching. However, the search step size δ_m updates quickly, that is to say, the BAS algorithm may fall into the local minimum due to converging prematurely. To solve this problem, an activation factor c_m is introduced. It enables the BAS algorithm to search larger space in the original step search rule and avoid falling into the local minimum prematurely at the same time. For readers' convenience, we introduce a unified vector $C(c_m, \zeta_m, \delta_m, \Delta W)$, which is used to control the searching update rule. The formula is defined as follows:

$$C(c_m, \zeta_m, \delta_m, \Delta W) = \begin{bmatrix} C_1(c_m, \Delta W) \\ C_2(\zeta_m, \Delta W) \\ C_3(\delta_m, \Delta W) \end{bmatrix}, \tag{7}$$

thereinto,

$$C_1(c_m, \Delta W) = \begin{cases} c_{m+1} = \varepsilon c_m & \text{if } |\Delta W| < c_m \\ c_{m+1} = c_m & \text{otherwise,} \end{cases} \tag{8}$$

$$C_2(\zeta_m, \Delta W) = \begin{cases} \zeta_{m+1} = 0.95\zeta_m + 0.01 & \text{if } |\Delta W| < c_m \\ \zeta_{m+1} = \zeta_m & \text{otherwise,} \end{cases} \tag{9}$$

$$C_3(\delta_m, \Delta W) = \begin{cases} \delta_{m+1} = 0.95\delta_m & \text{if } |\Delta W| < c_m \\ \delta_{m+1} = \delta_m & \text{otherwise.} \end{cases} \tag{10}$$

Remark 2. *The gradient estimate measure ΔW is used to compare with the activated threshold c_m to update ζ_m and δ_m. The larger the deviation between the particle and the target, the greater the value of ΔW. Thus, the search rule is updated when the value of ΔW is less than that of c_m, to explore the search area in large steps before finding the right direction. Thus, particles can fully explore the entire search area and find the global minimum value.*

Finally, the procedure of the NABAS algorithm is shown in Algorithm 1.

Remark 3. *At present, BAS and its variants have been deeply studied, and have practical applications in a wide range of fields. In [28], BAS was used to optimize the parameters of the combined motor in the power plant, which can effectively generate the optimal electric energy output. Zheng et al., discovered that BAS has good real-time performance, which provides a good solution for performance optimization control (PSC) to find the best engine performance for a specific flight mission [29]. Meanwhile, compared with commonly used machine learning methods, BAS and its variants have very good performance when applied to prediction [30,31]. Therefore, as an improved BAS, NABAS also can be applied in the same field.*

Algorithm 1: Nonlinear Activated Beetle Antennae Search (NABAS).

Loss Function: Define loss function (4)
Initialization : $\zeta \leftarrow 1, \delta \leftarrow 2, c \leftarrow 3, \varepsilon \leftarrow 0.95, k \leftarrow 150$

Generate random position $\mathbf{p}$;
$\mathbf{p}_0 \leftarrow \mathbf{p}$;
$W_0 \leftarrow W(\mathbf{p})$;
$\mathbf{p}_{\text{best}} \leftarrow \mathbf{p}_0$;
$W_{\text{best}} \leftarrow W_0$;
for $m \leftarrow 1$ **to** k **do**
    $\mathbf{u} \leftarrow$ random vector;
    $\mathbf{p}_r \leftarrow \mathbf{p}_m + \zeta_m \times \mathbf{u}$;
    $\mathbf{p}_l \leftarrow \mathbf{p}_m - \zeta_m \times \mathbf{u}$;
    $W_r \leftarrow W(\mathbf{p}_r)$;
    $W_l \leftarrow W(\mathbf{p}_l)$;
    $\Delta W \leftarrow W(\mathbf{p}_r) - W(\mathbf{p}_l)$;
    $\mathbf{p}_{m+1} \leftarrow \mathbf{p}_m - \delta_m \times \text{sign}[\Delta W] \times \mathbf{u}$;
    $W_{m+1} \leftarrow W(\mathbf{p}_{m+1})$;
    if $W_{m+1} < W_{\text{best}}$ **then**
        $W_{\text{best}} \leftarrow W_{m+1}$;
        $\mathbf{p}_{\text{best}} \leftarrow \mathbf{p}_{m+1}$;
    end
    if $\Delta W < c_m$ **then**
        $c_{m+1} \leftarrow \varepsilon c_m$;
        $\zeta_{m+1} \leftarrow 0.95\zeta_m + 0.01$;
        $\delta_{m+1} \leftarrow 0.95\delta_m$;
    end
end

4. Experiments and Verifications

Firstly, preparations of the dataset and hyperparameters are done for this problem, and the NABAS algorithm is used to solve the optimization problem (5). Then, the NABAS algorithm is compared with the other three methods, i.e., RUSBoost, logistics regression model, and SVM-FK.

4.1. Preparation

4.1.1. Preparation for Dataset

In Table 1, we enumerate the annual distribution of fraudulent companies in the dataset from 1979 to 2014. It can be seen that, compared with other years, the number of fraudulent companies accounted for a large proportion from 1991 to 2008. Considering the following four reasons, we choose the dataset from 1991 to 2008.

(1) Before 1991, not only was the number of fraudulent companies relatively low, but also some of the fraud activities were covered up. Since 1991, the nature of fraud had a significant shift in the United States. Then, the power and the law enforcement of the SEC were weakened [32].

(2) After 2008, the regulatory authorities began to reduce the law enforcement of accounting fraudulent companies, and the public resources used to detect fraudulent companies were greatly reduced. Therefore, fewer fraudulent companies were detected [32].

(3) Other technologies and relevant data we compared to our algorithm were quoted from the existing literature of other researchers, the dataset from 1991 to 2008 were used in their experiments. To increase the comparability of experiments, we used the data of the same year with them.

(4) The main contribution is to propose a universal model for the fraud detection. This model is insensitive to datasets.

In this way, the dataset from the year 1991 to 2008 is divided from Table 1. It contains 28 raw accounting variables for the training and testing of this model. In order to reduce the influence caused by the unbalanced characteristics on the experimental results, we first use SMOTE technology to oversample the minority class of fraudulent companies. Furthermore, to maintain the consistency of dataset distribution, six training sets are divided based on the year since 2001, and the datasets with an interval of two years are used as the test sets. The divided result is shown as Table 2.

Table 2. Training data and test data.

Year of Training Data	Training Data Size	Year of Test Data	Test Data Size
1991–2001	67,166	2003	5371
1991–2002	73,233	2004	5292
1991–2003	79,214	2005	5199
1991–2004	85,148	2006	5136
1991–2005	91,011	2007	5056
1991–2006	96,919	2008	4196

4.1.2. Preparation for Hyperparameters

It is significant to select properly hyperparameters, which helps the detection obtain an efficient result. For this fraud detection problem, 29 variables are used as the decision variables of NABAS, which contains 28 raw accounting variables and 1 bias b. Afterward, the setting of the hyperparameters (like $d = 1$) is the same as the original BAS algorithm, but δ is set as 2 to improve the convergence quality of the NABAS algorithm. Particularly, other hyperparameters of the model are determined by multiple experiments. For example, β controls the probability of whether a company is fraudulent or not, setting $\beta \leq 0.5$ to make the classification function more inclined to the category of fraud. For choosing a suitable value of β, we test β on the training and validation dataset of six periods. Then, we compute the average results of six experiments and show them in Table 3. Finally, by comparing the results, $\beta = 0.1$ is chosen as the threshold because of its good performance. In addition, we try different groups of (λ_1, λ_2) and test the performance of this model. The corresponding results are displayed in Table 4. It can be seen from Table 4 that, when the group of (λ_1, λ_2) is (1,5), the model achieves the best performance. For readers' convenience, all the hyperparameters of this problem are determined and separately shown in Table 5.

Table 3. Selection of hyperparameter β.

β	Sensitivity (%)	Precision (%)	NDCG
0.1	5.411	8.933	0.202
0.2	3.438	5.731	0.128
0.3	2.400	3.850	0.068
0.4	1.858	2.604	0.050
0.5	2.050	3.210	0.023

Table 4. Selection of hyperparameter (λ_1, λ_2).

(λ_1, λ_2)	Sensitivity (%)	Precision (%)	NDCG
(10,1)	2.941	7.407	0.170
(5,1)	2.941	7.407	0.152
(2,1)	3.677	9.259	0.184
(1,1)	1.471	3.704	0.090
(1,2)	5.147	12.963	0.278
(1,5)	8.088	20.370	0.355
(1,10)	5.882	14.815	0.291

Table 5. Values of hyperparameters.

NABAS		Model	
Hyperparameter	**Value**	**Hyperparameter**	**Value**
ζ	1	β	0.1
δ	2	λ_1	1
c	3	λ_2	5
ε	0.95		

4.2. Experiment Results

In this subsection, the performance of the NABAS algorithm is analyzed in detecting fraudulent companies on the given dataset from two aspects.

4.2.1. Convergence of Lost Function Value

From the above subsection, the entire dataset is divided into six subsets, and the percentage of fraudulent companies varies in each subset. Then, the NABAS is used for the six subsets. In order to show the convergence results of the loss function more completely, the maximum number of iterations k of the NABAS algorithm is set to 150. All convergence results of the loss function are shown in Figure 2. As can be seen from the figures, the loss function can achieve convergence to the optimal value at a fast speed, and the number of convergence iterations is less than 100.

4.2.2. Yearly Comparison with RUSBoost Algorithm

The RUSBoost algorithm has been compared with Logistics and SVM-FK in [23], and the results show that RUSBoost has the best performance. Therefore, we only compare NABAS with RUSBoost by year in four evaluation indicators: accuracy, sensitivity, precision, and NDCG. The corresponding results are presented in Table 6. As shown in Table 6, for sensitivity, although NABAS is slightly inferior in the test set 2003 and the test set 2004, with the increased number of training sets and test sets, the sensitivity of NABAS is higher than RUSBoost. For other indicators, the NABAS algorithm not only has obvious advantages in accuracy, precision, and NDCG but also remains in a relatively stable range of values. The NABAS algorithm shows better and more stable prediction ability than RUSBoost. In addition, the time indicator also plays a key role in the fraud detection problem. For the same test set, the time consumed by NABAS is much less than that spent by RUSBoost. As a consequence, the NABAS algorithm has better computational efficiency than RUSBoost.

Table 6. Comparison of NABAS and RUSBoost.

Training Set	Test Set	Method	Accuracy (%)	Sensitivity (%)	Precision (%)	NDCG (s)	t
1991–2001	2003	NABAS	**0.861**	5.882	**14.815**	**0.281**	**61.9**
		RUSBoost [23]	0.737	**10.14**	11.67	0.130	130.7
1991–2002	2004	NABAS	**0.853**	6.250	**13.208**	**0.271**	**70.0**
		RUSBoost [23]	0.772	**8.62**	8.47	0.079	149.5
1991–2003	2005	NABAS	**0.850**	**4.494**	**7.692**	**0.193**	**73.8**
		RUSBoost [23]	0.736	4.440	3.390	0.039	221.2
1991–2004	2006	NABAS	**0.854**	**4.546**	**5.882**	**0.149**	**79.3**
		RUSBoost [23]	0.762	3.030	1.690	0.019	256.5
1991–2005	2007	NABAS	**0.850**	**5.172**	**5.882**	**0.150**	**91.0**
		RUSBoost [23]	0.708	3.330	2.690	0.019	233.2
1991–2006	2008	NABAS	**0.858**	**6.122**	**6.122**	**0.168**	**93.0**
		RUSBoost [23]	0.641	0.000	0.000	0.000	235.9

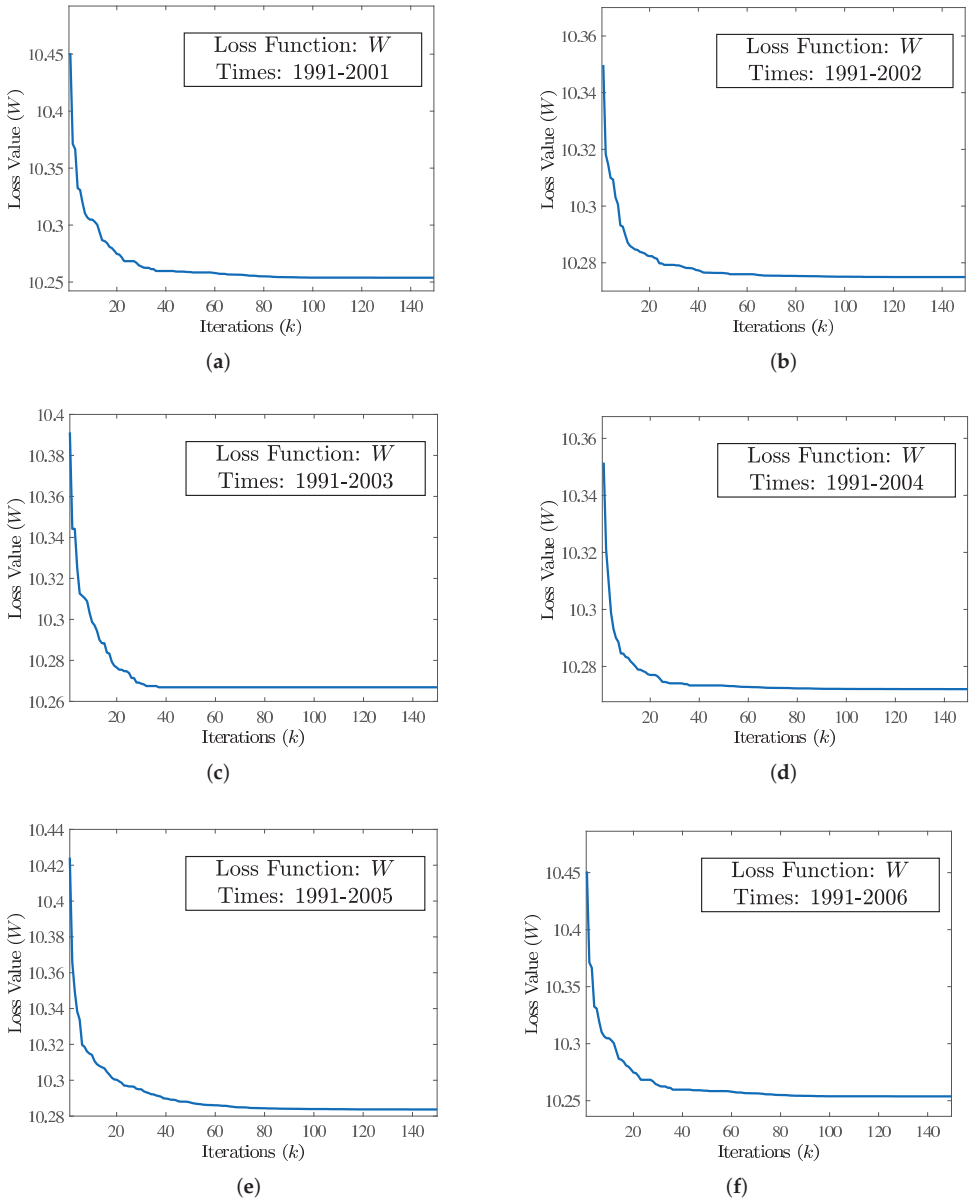


Figure 2. The convergence of loss function (W) equipped with NABAS on six subsets of training and validation datasets from 1991–2006. (**a**) Convergence of NABAS on training dataset 1991–2001. (**b**) Convergence of NABAS on training dataset 1991–2002. (**c**) Convergence of NABAS on training dataset 1991–2003. (**d**) Convergence of NABAS on training dataset 1991–2004. (**e**) Convergence of NABAS on training dataset 1991–2005. (**f**) Convergence of NABAS on training dataset 1991–2006.

4.2.3. Comparison with Other Public Algorithms

To further verify the performance of NABAS, logistic regression model [21] and SVM-FK [22], which proposed for fraud detection in recent years, are also used to be compared with NABAS. These methods are applied to datasets from 2003 to 2008, and the average results of all indicators are depicted in Table 7. From Table 7, one can learn that the NABAS algorithm is superior to the other three methods in all indicators.

Table 7. Comparison with other algorithms.

Variables of Dataset	Method	Accuracy (%)	Sensitivity (%)	Precision (%)	NDCG (s)	t
14 Financial Ratios	Logistic [21]	0.672	3.990	2.630	0.028	–
28 Raw Variables	Logistic [21]	0.690	0.730	0.850	0.006	–
	SVM-FK [22]	0.626	2.530	1.920	0.020	–
	RUSBoost [23]	0.725	4.880	4.480	0.049	204.5
	NABAS	**0.854**	**5.411**	**8.933**	**0.202**	**78.2**

5. Conclusions

This paper tackles the fraud detection problem of the American publicly traded companies, and uses a dataset that was collected from SEC's AAERs. For this problem, a fraud detection model has been proposed and transformed into an optimization problem of minimizing loss function. Meanwhile, the NABAS algorithm has been presented and employed to solve this optimization problem. Furthermore, three methods have also been used to compare with the NABAS algorithm. The experiment results show that the NABAS algorithm has better performance than other methods in all indicators. The advantage of this paper lies in the introduction of NABAS to solve the fraud detection problem and obtain higher accuracy and efficiency. However, in this study, the parameter selection is manual, which depends on repeated experiments and will incur a huge time cost. Therefore, developing an algorithm for parameter selection is a future research direction of the work.

Author Contributions: Conceptualization, B.L. and J.L.; methodology, J.L. and B.L.; software, Z.H. and X.C.; validation, Z.H. and B.L.; formal analysis, Z.H.; investigation, X.C. and Z.H.; data curation, X.C. and Z.H.; writing—original draft preparation, Z.H.; writing—review and editing, B.L. and Z.H.; visualization, Z.H.; supervision, B.L. and J.L.; project administration, B.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grants 62066015 and 61962023, the Natural Science Foundation of Hunan Province of China under Grant 2020JJ4511, and the Research Foundation of Education Bureau of Hunan Province, China, under Grant 20A396.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alghofaili, Y.; Albattah, A.; Rassam, M.A. A Financial Fraud Detection Model Based on LSTM Deep Learning Technique. *J. Appl. Secur. Res.* **2020**, *15*, 498–516. [CrossRef]
2. Delecourt, S.; Guo, L. Building a robust mobile payment fraud detection system with adversarial examples. In Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 3–5 June 2019; pp. 103–106.
3. Peng, L.; Lin, R. Fraud phone calls analysis based on label propagation community detection algorithm. In Proceedings of the 2018 IEEE World Congress on Services (SERVICES), San Francisco, CA, USA, 2–7 July 2018; pp. 23–24.
4. Lin, T.H.; Jiang, J.R. Credit Card Fraud Detection with Autoencoder and Probabilistic Random Forest. *Mathematics* **2021**, *9*, 2683. [CrossRef]
5. Ngai, E.W.; Hu, Y.; Wong, Y.H.; Chen, Y.; Sun, X. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decis. Support Syst.* **2011**, *50*, 559–569. [CrossRef]
6. Hilal, W.; Gadsden, S.A.; Yawney, J. A Review of Anomaly Detection Techniques and Applications in Financial Fraud. *Expert Syst. Appl.* **2021**, *193*, 116429. [CrossRef]
7. Reurink, A. Financial fraud: A literature review. *J. Econ. Surv.* **2018**, *32*, 1292–1325. [CrossRef]

8. Chandradeva, L.S.; Jayasooriya, I.; Aponso, A.C. Fraud Detection Solution for Monetary Transactions with Autoencoders. In Proceedings of the 2019 National Information Technology Conference (NITC), Colombo, Sri Lanka, 8–10 October 2019; pp. 31–34.
9. Jain, V.; Agrawal, M.; Kumar, A. Performance Analysis of Machine Learning Algorithms in Credit Cards Fraud Detection. In Proceedings of the 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), Noida, India, 4–5 June 2020; pp. 86–88.
10. Yao, L.; Wong, P.K.; Zhao, B.; Wang, Z.; Lei, L.; Wang, X.; Hu, Y. Cost-Sensitive Broad Learning System for Imbalanced Classification and Its Medical Application. *Mathematics* **2022**, *10*, 829. [CrossRef]
11. Lemnaru, C.; Potolea, R. Imbalanced classification problems: Systematic study, issues and best practices. In *International Conference on Enterprise Information Systems*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 35–50.
12. Jiang, X.; Li, S. BAS: Beetle Antennae Search Algorithm for Optimization Problems. *arXiv* **2017**, arXiv: abs/1710.10724.
13. Zhang, Y.; Li, S.; Xu, B. Convergence analysis of beetle antennae search algorithm and its applications. *Soft Comput.* **2021**, *25*, 10595–10608. [CrossRef]
14. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying minimum-cost portfolio insurance under transaction costs problem via Beetle Antennae Search Algorithm (BAS). *Appl. Math. Comput.* **2020**, *385*, 125453. [CrossRef]
15. Katsikis, V.N.; Mourtas, S.D. Binary beetle antennae search algorithm for tangency portfolio diversification. *J. Model. Optim.* **2021**, *13*, 44–50. [CrossRef]
16. Katsikis, V.N.; Mourtas, S.D. Optimal portfolio insurance under nonlinear transaction costs. *J. Model. Optim.* **2020**, *12*, 117–124. [CrossRef]
17. Medvedeva, M.A.; Katsikis, V.N.; Mourtas, S.D.; Simos, T.E. Randomized time-varying knapsack problems via binary beetle antennae search algorithm: Emphasis on applications in portfolio insurance. *Math. Methods Appl. Sci.* **2021**, *44*, 2002–2012. [CrossRef]
18. Mourtas, S.D.; Katsikis, V.N. V-Shaped BAS: Applications on Large Portfolios Selection Problem. In *Computational Economics*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 1–21.
19. Khan, A.T.; Cao, X.; Li, S.; Hu, B.; Katsikis, V.N. Quantum beetle antennae search: A novel technique for the constrained portfolio optimization problem. *Sci. China Inf. Sci.* **2021**, *64*, 1–14. [CrossRef]
20. Khan, A.T.; Cao, X.; Li, S.; Katsikis, V.N.; Brajevic, I.; Stanimirovic, P.S. Fraud detection in publicly traded US firms using Beetle Antennae Search: A machine learning approach. *Expert Syst. Appl.* **2022**, *191*, 116148. [CrossRef]
21. Dechow, P.M.; Ge, W.; Larson, C.R.; Sloan, R.G. Predicting material accounting misstatements. *Contemp. Account. Res.* **2011**, *28*, 17–82. [CrossRef]
22. Cecchini, M.; Aytug, H.; Koehler, G.J.; Pathak, P. Detecting management fraud in public companies. *Manag. Sci.* **2010**, *56*, 1146–1160. [CrossRef]
23. Bao, Y.; Ke, B.; Li, B.; Yu, Y.J.; Zhang, J. Detecting accounting fraud in publicly traded US firms using a machine learning approach. *J. Account. Res.* **2020**, *58*, 199–235. [CrossRef]
24. Lasisi, A.; Attoh-Okine, N. Principal components analysis and track quality index: A machine learning approach. *Transp. Res. Part C: Emerg. Technol.* **2018**, *91*, 230–248. [CrossRef]
25. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
26. Ruuska, S.; Hämäläinen, W.; Kajava, S.; Mughal, M.; Matilainen, P.; Mononen, J. Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. *Behav. Process.* **2018**, *148*, 56–62. [CrossRef] [PubMed]
27. Wang, Y.; Wang, L.; Li, Y.; He, D.; Chen, W.; Liu, T.Y. A theoretical analysis of NDCG ranking measures. In Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013), Citeseer, Princeton, NJ, USA, 12–14 June 2013; pp. 1–26.
28. Ghosh, T.; Martinsen, K.; Dan, P.K. Data-Driven Beetle Antennae Search Algorithm for Electrical Power Modeling of a Combined Cycle Power Plant. In *World Congress on Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 906–915.
29. Zheng, Q.; Xiang, D.; Fang, J.; Wang, Y.; Zhang, H.; Hu, Z. Research on performance seeking control based on beetle antennae search algorithm. *Meas. Control* **2020**, *53*, 1440–1445. [CrossRef]
30. Sun, Y.; Zhang, J.; Li, G.; Wang, Y.; Sun, J.; Jiang, C. Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes. *Int. J. Numer. Anal. Methods Geomech.* **2019**, *43*, 801–813. [CrossRef]
31. Xie, S.; Chu, X.; Zheng, M.; Liu, C. Ship predictive collision avoidance method based on an improved beetle antennae search algorithm. *Ocean Eng.* **2019**, *192*, 106542. [CrossRef]
32. Rakoff, J.S. The financial crisis: Why have no high-level executives been prosecuted? *N. Y. Rev. Books* **2014**, *9*, 7.

MDPI AG
Grosspeteranlage 5
4052 Basel
Switzerland
Tel.: +41 61 683 77 34

Mathematics Editorial Office
E-mail: mathematics@mdpi.com
www.mdpi.com/journal/mathematics



Disclaimer/Publisher's Note: The title and front matter of this reprint are at the discretion of the Guest Editors. The publisher is not responsible for their content or any associated concerns. The statements, opinions and data contained in all individual articles are solely those of the individual Editors and contributors and not of MDPI. MDPI disclaims responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.





Academic Open
Access Publishing

mdpi.com

ISBN 978-3-7258-7463-7