ЕКОНОМІКО-МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ БІЗНЕСОВИХ ПРОЦЕСІВ

UDC 336.712.38
JEL Classification: G11, B50
DOI: https://doi.org/10.20535/2307-5651.34.2025.341984

Lazarenko Iryna
Ph.D. in Mathematics
(corresponding author)
ORCID ID: 0000-0002-3384-1186

Krykun Yevhen
Master Student
ORCID ID: 0009-0001-5146-4273
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

PORTFOLIO MANAGEMENT WITH DATA MINING TECHNIQUES IN TIME SERIES ANALYSIS

This study explores the application of Data Mining techniques, specifically deep neural networks (DNN) and recurrent neural networks (RNN), for optimizing stock portfolios. Using time series data, we compare the performance of DNN and RNN models in predicting stock prices and constructing optimal portfolios. Key evaluation metrics demonstrate that the RNN model's forecasts yield a portfolio with income and risk metrics that closely match actual values, outperforming the DNN model. Furthermore, the RNN model's portfolio weights show a stronger alignment with actual distributions, indicating superior predictive accuracy in asset allocation. This study concludes that RNN, with their inherent capability for processing sequential data, are particularly well-suited for time series forecasting in financial applications.

Keywords: *portfolio management, Data Mining techniques, Deep Neural Networks, Recurrent Neural Networks, investment.*

Лазаренко І. С., Крикун Є. О.
Національний технічний університет України
«Київський політехнічний університет імені Ігоря Сікорського»

УПРАВЛІННЯ ПОРТФЕЛЕМ ЦІННИХ ПАПЕРІВ З ВИКОРИСТАННЯМ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ У АНАЛІЗІ ЧАСОВИХ РЯДІВ

У даній статті комплексно досліджується застосування передових методів інтелектуального аналізу даних (Data Mining), зокрема глибоких нейронних мереж (DNN) та рекурентних нейронних мереж (RNN), для оптимізації управління інвестиційними портфелями. Незважаючи на теоретичну цінність класичних портфельних теорій, їхня неефективність у умовах швидкозмінних ринків обумовлює необхідність розробки сучасних підходів до прогнозування фондових ринків. Дане дослідження ґрунтується на аналізі часових рядів із застосуванням порівняльного підходу до оцінки ефективності моделей DNN та RNN у прогнозуванні динаміки цін акцій та подальшій оптимізації структури інвестиційного портфеля. Результати дослідження, отримані на основі ключових метрик якості (MSE, MAE, MAPE,), демонструють перевагу моделі RNN, яка системно показує кращу відповідність реальним ринковим даним у порівнянні з DNN. Прогнозні моделі, побудовані на основі RNN, дозволяють формувати портфель з оптимальними співвідношеннями доходу та ризику, максимально наближеними до реальних ринкових умов, а також забезпечують більш точний розподіл ваг активів. Це свідчить про вищу прогностичну точність RNN у задачах розподілу активів та управління портфелем. Можна зробити висновок, що RNN, завдяки архітектурі, орієнтованій на обробку послідовних даних, є перспективнішим інструментом для прогнозування фінансових часових рядів. Отримані результати підкреслюють потенціал методів інтелектуального аналізу даних для вдосконалення інвестиційних стратегій та обґрунтовують необхідність подальших досліджень щодо розробки гібридних моделей для оптимізації портфелів у умовах високої ринкової волатильності. Окрім того, в статті розглядаються практичні аспекти імплементації цих моделей у реальні системи торгівлі, аналізуються обмеження, пов'язані з їхнім застосуванням, та пропонуються напрями подальших досліджень для подолання цих обмежень. Дослідження також висвітлює важливість обробки великих масивів даних реального часу та необхідність адаптації моделей до різних ринкових режимів для забезпечення стабільної ефективності.

Ключові слова: *управління інвестиційним портфелем, методи інтелектуального аналізу даних, глибокі нейронні мережі, рекурентні нейронні мережі, інвестування.*

© Lazarenko Iryna, Krykun Yevhen, 2025

Problem statement. In the context of an active war phase, the question of investing funds in foreign securities has become increasingly relevant for Ukrainian investors. One of the most attractive and relatively low-risk options is the U.S. stock market, which has demonstrated resilience even during global financial crises. Constructing a stock portfolio is a complex process that requires a broad skill set, encompassing not only portfolio theory but also an understanding of stock market trends and the impact of geopolitics and economic shifts on the global economy.

While traditional methods for constructing an optimal portfolio date back to the 1950s and have been extensively studied, these approaches have limitations due to their inability to fully account for current trends and rapid changes in the economy. This research aims to explore a modernized approach to portfolio optimization by applying Data Mining techniques in time series analysis. Contemporary Data Mining methods, such as Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN), enhance the accuracy of stock predictions, demonstrating significantly higher precision than classical algorithms like ARIMA. These models not only improve forecasting but also allow for portfolio construction across different timeframes, including forecasts for several months into the future.

This study conducts a comparative analysis of two types of neural networks: DNN and RNN. This comparison is somewhat unconventional, as stock price forecasting typically utilizes another type of recurrent neural network – Long Short-Term Memory (LSTM). Based on the predicted values, a stock portfolio will be constructed, and the forecasted outcomes will be compared with actual portfolio performance. This analysis will help determine which method performs better on a selected set of corporate securities, providing insights into the efficacy of Data Mining techniques in portfolio management.

Analysis of recent research and publications. The literature review should begin with an examination of classical portfolio theory, pioneered by the renowned American economist Harry Markowitz. American economist Harry Markowitz was one of the first to recognize the advantages of optimizing a stock portfolio, writing a dissertation on "Portfolio Selection" in 1952. Modern Portfolio Theory (MPT) remains a widely adopted investment strategy that contrasts with traditional stock picking. MPT provides tools for portfolio management that, when applied correctly, can help create a diversified and profitable investment portfolio [1].

MPT is a theory of financial investments that employs statistical methods to optimize risk distribution in a securities portfolio and to estimate returns. Key components of this theory include asset valuation, investment decision-making, portfolio optimization, and performance measurement. Despite its abstract nature and lack of consideration for practical aspects like taxes and operational costs, as well as assumptions about infinite divisibility of assets and uniform investor information, MPT and its advancements in CAPM and arbitrage theory have significant practical value. These models offer foundational insights into balancing portfolio returns and risk [2].

On an ideal securities market, portfolio managers can analyze market trends, forecast future performance, and evaluate the investment characteristics of financial instruments. Theoretical aspects of MPT reveal general patterns in the securities market, enabling effective criteria for practical application. MPT assumes that in developed financial markets, institutions are well-informed and operational costs are negligible relative to transaction volumes, allowing these factors to be disregarded in certain cases [2].

Markowitz's model presents portfolio formation as a combination of potential investments, with the primary goal being to find optimal asset allocation proportions that minimize risk for a given level of return or maximize expected returns at an acceptable risk level. The model identifies an "efficient" portfolio, which offers the least risk at a specified return level, but it does not suggest a single optimal portfolio.

These theoretical developments are crucial for investors, allowing a rational approach to building a stock portfolio that considers risk levels and expected returns. An optimal portfolio helps diversify risks and maximize potential returns. Additionally, Markowitz's model and MPT enable investors to make informed decisions and effectively manage their investments.

Key assumptions of the Markowitz model include:

1. Expected returns are represented by the mathematical expectation of returns.

2. Risk is represented by the standard deviation of returns.

3. Past data used in return and risk calculations fully reflect future values.

4. The correlation coefficient expresses the degree and nature of relationships between securities [3].

Markowitz's formula allows investors to mathematically align risk tolerance with reward expectations, forming an ideal portfolio. This theory is based on two core principles:

1. Every investor aims to maximize return at any risk level.

2. Risk can be reduced by diversifying a portfolio with uncorrelated securities.

In stock market prediction, accurate forecasting of stock prices is highly valuable for investors and analysts. Data Mining techniques, especially Dense Neural Networks (DNNs) and Simple Recurrent Neural Networks (RNNs), have been widely applied for time series forecasting due to their ability to model complex, non-linear patterns inherent in financial data. Both methods offer distinct advantages in processing and forecasting stock prices, with DNNs excelling in capturing static patterns and RNNs in handling sequential data and temporal dependencies.

DNNs, also known as fully connected networks, consist of multiple layers of interconnected neurons. Each neuron in a layer is connected to every neuron in the preceding and succeeding layers, enabling the model to capture patterns by learning non-linear transformations through its layers. DNNs are advantageous for tasks where relationships between features are non-linear and complex, which aligns well with stock market data's inherent unpredictability [4].

In stock prediction, DNNs are often applied to transform historical stock prices, trading volumes, and various financial indicators into useful feature representations. By learning from historical data, a DNN can generalize patterns such as the impact of trading volume on stock price trends or the correlation between prices of different stocks. For example, a model trained with price movements over the last year may capture essential patterns indicating growth, stability, or volatility, which aids in predicting future stock behaviour.

One limitation of DNNs is their lack of inherent sequential memory, which makes it challenging to capture time dependencies. Stock prices, however, rely heavily on historical prices and events, often demanding a model with memory to retain information about the sequence of past observations. This limitation makes DNNs more suitable for static features or scenarios where past trends are assumed to have limited long-term influence.

Simple RNNs are a natural fit for time series forecasting due to their ability to maintain a sequential understanding of data over time. Unlike DNNs, RNNs are designed with loops within their architecture, allowing them to "remember" previous outputs by storing information about past inputs. This memory-like structure is well-suited for tasks that involve temporal dependencies, such as predicting future stock prices based on historical price sequences.

In stock prediction, RNNs are trained on sequences of past stock prices, and they learn patterns that can inform the future trajectory of the price. This includes identifying trends, such as cyclical price patterns or momentum, which are characteristic of stock market behavior. For instance, an RNN model can learn from past upswings and downswings in stock prices, adjusting predictions based on recent patterns rather than treating each data point in isolation. Such a method enables the model to forecast prices more effectively by taking advantage of the correlation across different time steps.

However, Simple RNNs are not without limitations. One primary drawback is their tendency to suffer from vanishing gradient issues, which can lead to the model "forgetting" information from earlier time steps. This problem can affect prediction accuracy, particularly when forecasting based on long time series, which is typical in stock market analysis. Advanced variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks have been introduced to mitigate this issue, though Simple RNNs are still useful in shorter-term or smaller-scale time series prediction tasks [4].

Formulating the purposes of the article. This research aims to develop a modernized portfolio optimization framework that enhances the classical Markowitz theory through the integration of forward-looking risk assessment methodologies. The primary objective is to address the inherent limitation of traditional models, which rely exclusively on ex-post factual data and consequently fail to adequately incorporate prevailing market trends and future dynamics.

To achieve this objective, the study formalizes a multicriteria optimization problem that simultaneously maximizes expected return and minimizes portfolio risk by converging these dual objectives into a single minimization functional subject to constraints. A crucial scientific contribution involves the proposed substitution of the historical covariance matrix with its forecasted counterpart, enabling more accurate quantification of future stochastic dependencies among assets.

Leveraging contemporary big data analytics capabilities, the proposed approach facilitates the construction of more adaptive and robust portfolios capable of responding to evolving market conditions, rather than merely reflecting their historical behavior. This methodological advancement represents a significant departure from conventional portfolio optimization techniques by incorporating probabilistic forward-looking measures into the risk-return paradigm.

Presentation of the main research material. In accordance with Markowitz's theory, the expected return of a portfolio is calculated by a formula that integrates various factors and their interrelations:

$$R_p = \sum_i R_i w_i, \tag{1}$$

where R_p is the return on the portfolio,

R_i – return on the asset,

w_i – share of the asset in the portfolio.

The portfolio's expected risk quantifies its potential for loss, indicating the likelihood of capital reduction based on the chosen asset allocation and weights. The expected portfolio risk is mathematically represented as the standard deviation of its return. Prior to obtaining this standard deviation, we first calculate the return variance, which measures the dispersion or fluctuation in the returns of assets. A higher variance implies a greater volatility risk, as actual returns may diverge from expectations. In Markowitz's framework, this variance is defined by:

$$\sigma_p^2 = \sum_i \sum_j w_i w_j cov\left(R_i, R_j\right) = \sum_i \sum_j w_i w_j \sigma_{ij} \tag{2}$$

where σ_p^2 is the variance of portfolio returns.

The portfolio's expected risk, or standard deviation, is derived as:

$$\sigma_p = \sqrt{\sigma_p^2}, \tag{3}$$

This portfolio is constructed based on Markowitz's portfolio theory, incorporating specific modifications. The expected return and risk of the portfolio are calculated using formulas (1) to (3). For effective optimization, focusing on both return maximization and risk minimization, multicriteria optimization is applied.

The first criterion is the expected return function, optimized as follows:

$$f_1 = R_p \to max, \tag{4}$$

The second criterion is to minimize the total portfolio risk:

$$f_2 = \sigma_p \to min, \tag{5}$$

With both objective functions established, we proceed to a convolution of criteria, optimizing the objective function for a minimum while subject to the constraints:

$$\begin{cases} \sum_{i=1}^{n} w_i = 1 \\ w_i \geq 0{,}01, \\ R_p > 0 \\ \sigma_p \geq 0 \end{cases} \tag{6}$$

The target function that will be optimized:

$$W = \frac{\alpha * \sigma_p}{(1-\alpha) * R_p} \to min, \tag{7}$$

Having defined the main criterion and constraints, we have an optimization problem to find the optimal stock portfolio:

$$W = \frac{\alpha * \sqrt{w^T * cov * w}}{(1-\alpha) * \sum_{i=1}^{n} R_i * w_i} \to min$$

$$\begin{cases} \sum_{i=1}^{n} w_i = 1 \\ w_i \geq 0{,}01 \\ \sum_{i=1}^{n} R_i * w_i > 0 \\ \sqrt{w^T * cov * w} \geq 0 \end{cases}, \quad (8)$$

With the development of technologies and Big Data tools, Markowitz's theory can be easily applied in practice for a large set of different papers in a portfolio. Despite this, Markowitz's theory has a significant flaw – the lack of consideration for market trends. To overcome this flaw in Markowitz's theory, a modernized method of calculating the covariance matrix is applied, which is used for risk calculation – forecasting the covariance matrix of risks [5].

Deep Neural Networks with Dense Layers. A dense neural network, also called a fully connected or feed-forward neural network, is built by stacking several layers of neurons. In this structure, each layer is linked to all neurons in the previous layer, enabling the model to detect and learn data patterns. In a sequential model, data passes through each layer one after another, moving from input to output. The mathematical representation of a dense layer is:

$$y = f(Wx + b), \quad (9)$$

where W – weight matrix,
x – input vector,
b – bias vector,
f – activation function (in our case ReLU) [6].

The ReLU (Rectified Linear Unit) activation function outputs the input directly if it is positive; otherwise, it returns zero, allowing for faster training by introducing non-linearity while mitigating the vanishing gradient problem. The formula for the ReLU (Rectified Linear Unit) activation function is:

$$ReLU(x) = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x \, if \, x > 0 \\ 0 \, if \, x \leq 0 \end{cases}, \quad (10)$$

where x – the input to a neuron [7].

The output layer is a single neuron that provides the predicted stock price at the next time step. For regression tasks, the output neuron typically uses a linear activation function:

$$\hat{y} = z^{(L)} = \sum_{i=1}^{N_{L-1}} w_i^{(L)} z_i^{(L-1)} + b^{(L)}, \quad (11)$$

where $w_i^{(L)}$ – the weights of the output layer,
$b^{(L)}$ – the bias of the output layer,
$\hat{y}$ – the*y* predicted stock price for the next time step.

In Python, the Sequential model with dense layers is structured as follows.

The model followed by three dense (fully connected) hidden layers with ReLU activations and neuron counts of 150, 100, and 50, respectively. The output layer has a single neuron for the final prediction. The model is optimized with Stochastic Gradient Descent (SGD) with momentum, set at a learning rate of 10-6 and a momentum factor of 0.9, and is compiled with Mean Squared Error (MSE) as the loss function. It is trained over 150 epochs on the given dataset.

Recurrent Neural Networks. Recurrent layers, first introduced in the 1980s, form the foundational structure of Recurrent Neural Networks (RNNs), which are specifically engineered to handle sequential data. This design makes RNNs well-suited for applications in natural language processing, time series forecasting, and sequence-to-sequence learning. A notable early model is the Elman Network, developed by Jeff Elman. Recurrent layers incorporate an internal hidden state that evolves dynamically over time, enabling the network to retain information from previous time steps. The update of this hidden state is mathematically expressed as:

$$h_t = f(W_x h \cdot x_t + W_h h \cdot h(t-1) + b_h), \quad (12)$$

where h_t – hidden state at time step t,
x_t – input at time step t,
$W_x h$ – input-to-hidden weight matrix,
$W_h h$ – hidden-to-hidden weight matrix,
b_h – bias vector,
f – activation function (in our case ReLU) [6].

```
model_tune = tf.keras.models.Sequential([
  tf.keras.Input(shape=(window_size,)),
  tf.keras.layers.Dense(150, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(50, activation="relu"),
  tf.keras.layers.Dense(1)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)

model_tune.compile(loss="mse", optimizer=optimizer)

history = model_tune.fit(dataset, epochs=150, verbose=0)
```

Figure 1. DNN with Dense Layers structure

For the RNN model, the same activation function is used, which can be seen in formula (10). In Python, the RNN Sequential model with SimpleRNN layers is structured as follows.

The model consists of an input layer with a specified window size, followed by two SimpleRNN layers with 40 units each, using the ReLU activation function. The first RNN layer returns sequences, feeding into the second RNN layer, which connects to a final Dense layer with a single output neuron for regression. The model uses the Huber loss function and is optimized with Stochastic Gradient Descent (SGD) with a learning rate of 10-5 and momentum of 0.9, and it tracks Mean Absolute Error (MAE) as a performance metric. Training is conducted over 100 epochs on the provided dataset.

Evaluation Metrics. In time series analysis, four primary metrics are commonly employed to evaluate model performance: Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and the Coefficient of Determination (R^2).

MAE measures the average magnitude of errors between predicted and observed values, without regard to direction, providing a straightforward interpretation of the typical prediction error [8]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|, \quad (13)$$

where y_i – the observed value,

$\hat{y}_i$ – the predicted value,

n – the total number of observations.

MSE calculates the average of the squared differences between predicted and observed values, penalizing larger errors more heavily and highlighting significant discrepancies [8]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2, \quad (14)$$

MAPE expresses the average absolute error as a percentage of observed values, making it scale-independent and useful for comparing models across different datasets:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \quad (15)$$

R^2 quantifies the proportion of variance in the observed data that the model explains, indicating the model's overall fit, with values closer to 1 showing better explanatory power [8]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}, \quad (16)$$

Each of these metrics provides unique insights into model accuracy and prediction reliability.

DNNs Predictions. In the graph below, we see a comparison of predicted values with actual values for each time series.

The graphical analysis shows that the forecasting is quite accurate, with DNN models being good at repeating and understanding data trends. To confirm this, let's look at the key metrics for each time series. The results are presented in Table 1.

The models demonstrate a generally strong predictive performance. The Mean Squared Error (MSE) and Mean Absolute Error (MAE) vary across stocks, with MSFT and NVDA having higher MSE and MAE values, suggesting greater variance in prediction accuracy for these assets. In contrast, AAPL and V exhibit lower MSE and MAE, indicating more accurate predictions. The Mean Absolute Percentage Error (MAPE) is also relatively low across the board, with V achieving the lowest at 0.825, reflecting high prediction reliability in percentage terms. The R^2 values are consistently high (above 0.97 for all stocks), showing the model's strong ability to capture the variance in the data. Overall, while the DNN model performs well, prediction accuracy could be improved for stocks with higher MSE and MAE, such as MSFT and NVDA.

RNNs Predictions. The graph below shows a comparison between the predicted values and the actual values for each time series.

The graphical analysis shows that prediction by RNN models is quite accurate and shows better results than prediction by DNN models. To confirm this, let's display the key indicators in Table 2.

The Mean Squared Error (MSE) values indicate relatively low prediction error for most stocks, especially for GOOG (7.452) and V (8.606), while MSFT has a notably higher MSE (25.802), suggesting greater variance in accuracy. Mean Absolute Error (MAE) and Mean Absolute

```
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size, 1)),
    tf.keras.layers.SimpleRNN(40, return_sequences=True, activation='relu'),
    tf.keras.layers.SimpleRNN(40, activation='relu'),
    tf.keras.layers.Dense(1)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=1e-5, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])

history = model.fit(dataset,epochs=100, verbose=0)
```

Figure 2. RNN Sequential model

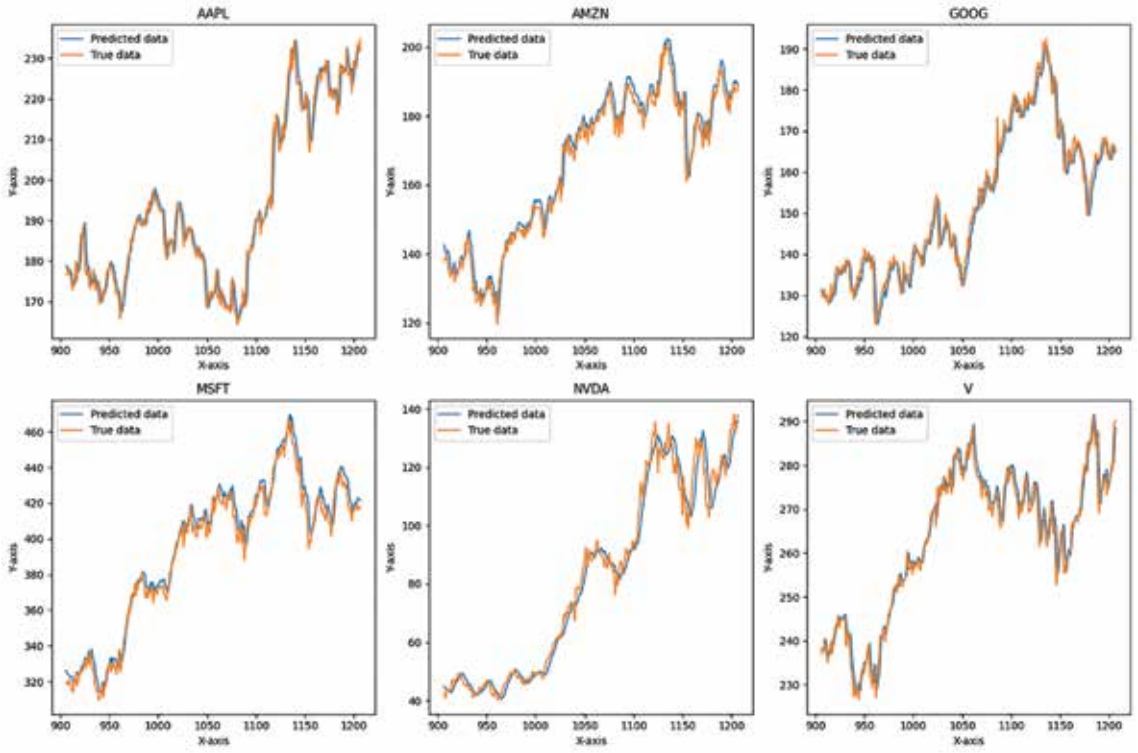

Figure 3. DNNs Predictions

Table 1

DNNs Predictions

	AAPL	AMZN	GOOG	MSFT	NVDA	V
MSE	8.939	12.586	9.078	37.687	21.084999	8.606
MAE	2.237	2.694	2.305	4.750	3.250000	2.161
MAPE	1.158	1.653	1.518	1.216	3.801000	0.825
R^2	0.979	0.973	0.970	0.977	0.980	0.969

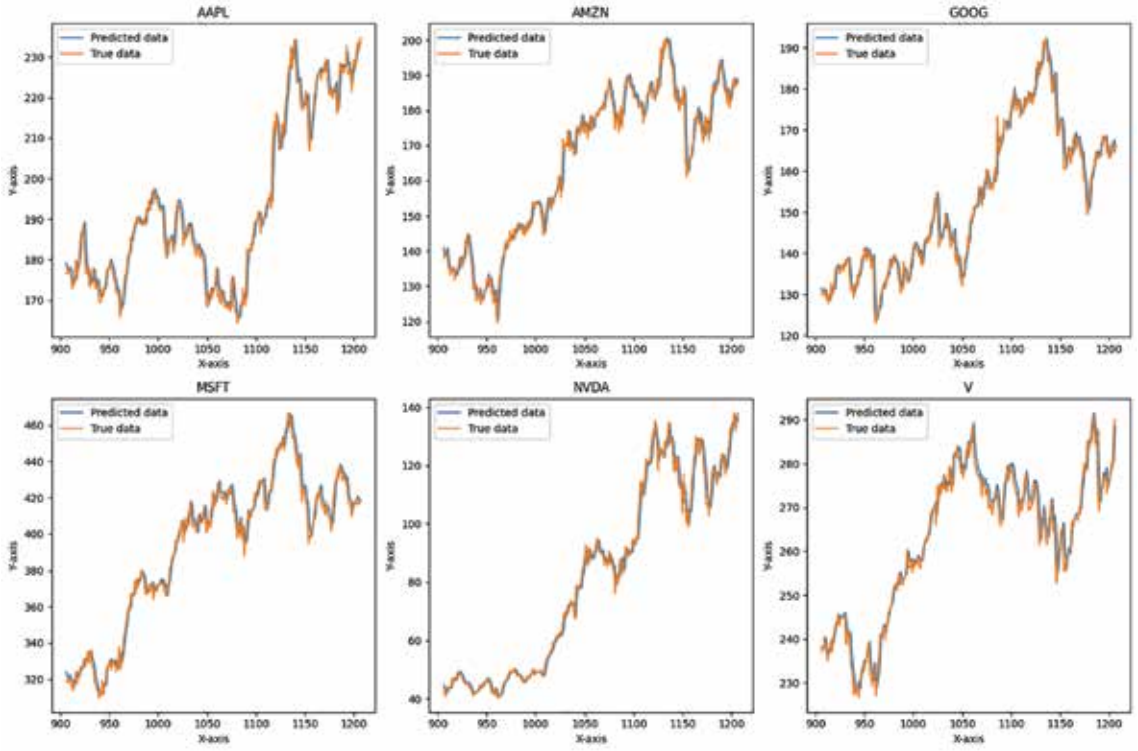

Figure 4. RNNs Predictions

Table 2

RNNs Predictions

	AAPL	AMZN	GOOG	MSFT	NVDA	V
MSE	9.041	9.356	7.452	25.802	10.213	8.606
MAE	2.227	2.264	1.915	3.932	2.221	2.161
MAPE	1.157	1.394	1.267	1.003	2.543	0.825
R^2	0.979	0.980	0.976	0.985	0.990	0.969

Percentage Error (MAPE) metrics are similarly consistent across stocks, with MSFT and NVDA showing slightly elevated values, reflecting less precise predictions for these assets. The R^2 values are very high, with all stocks exceeding 0.96, which demonstrates the RNN model's effectiveness in capturing the variance within the data and providing reliable forecasts. NVDA achieves the highest R^2 (0.990), suggesting excellent alignment between predicted and actual values. Overall, the RNN model displays strong forecasting performance across most stocks, with room for improvement in cases like MSFT, where prediction variance is higher.

Portfolio Building. For the final analysis of the results, we will build portfolios based on the predicted values of DNN, RNN, and on the actual values. We will present the results in the form of a comparative Table 3:

Based on these indicators, we have weighting factors for each portfolio compared to the actual weights:

Based on the results, the RNN model provides a more accurate prediction for portfolio construction. The RNN-predicted income 0.186 is closer to the actual income 0.191 than the DNN's prediction of 0.260, indicating a better alignment with real performance. Additionally, while both models estimate lower risk than the actual risk of 1.429, the RNN model's risk prediction 1.076 is slightly closer to the actual value compared to the DNN's risk prediction 1.016.

Table 3

Portfolio Results

	DNN	RNN	Actual
Income	0.260	0.186	0.191
Risk	1.016	1.076	1.429

Table 4

Portfolio Weights

	DNN	RNN	Actual
AAPL	0.010	0.082	0.148
AMZN	0.010	0.010	0.010
GOOG	0.010	0.010	0.010
MSFT	0.039	0.010	0.010
NVDA	0.641	0.393	0.370
V	0.290	0.495	0.452

In terms of portfolio weights, the RNN model's weights are generally closer to the actual distribution, particularly for V and NVDA. Therefore, the RNN model is the more accurate predictor overall.

Conclusions. This study presents a comparative analysis of two Data Mining techniques, Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN), for portfolio optimization.

The research findings reveal that both DNN and RNN models provide reliable forecasts for stock prices, as demonstrated by high R^2 values for all analyzed stocks, indicating each model's capacity to explain variance effectively. However, the RNN model consistently outperforms the DNN in terms of prediction accuracy across most metrics. The RNN model achieved lower Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) for several stocks. This superior performance may be attributed to RNN's sequential structure, which allows it to capture temporal dependencies essential in stock market data.

In the context of portfolio construction, RNN predictions also yield a portfolio with income and risk figures closer to the actual values than those derived from DNN predictions. The RNN-predicted income of 0.186 is closer to the actual income of 0.191, compared to the DNN's 0.260. Additionally, the RNN model's risk prediction (1.076) aligns more closely with the actual portfolio risk (1.429) than the DNN's 1.016. Furthermore, the RNN-generated portfolio weights, particularly for high-weight assets like V and NVDA, closely resemble the actual portfolio distribution, highlighting its superior predictive accuracy in asset allocation.

While both models contribute valuable insights into stock prediction and portfolio management, the RNN model demonstrates a stronger overall performance. The sequential learning capability of RNNs makes them better suited for time series analysis in financial forecasting. This study confirms the potential of Data Mining methods to enhance portfolio optimization by providing more accurate forecasts, enabling investors to make data-driven decisions with greater confidence. Future research could explore hybrid approaches, combining RNNs with more complex network types like LSTM or GRU, to further refine portfolio forecasting accuracy in volatile markets.

References:

1. Mazhara G. A., Krykun Y. O. Modeling of the optimal investment portfolio focused on risk minimization. *Modern Economics.* 2023. № 38(2023). P. 69–75. DOI: https://doi.org/10.31521/modecon.V38(2023)-11

2. 9.4. Use of modern portfolio theory in portfolio management. Capital asset pricing model (CAPM). The required rate of return. Arbitrage theory. Evaluation of the effectiveness of portfolio management. Performance criteria. Available at: https://buklib.net/books/26654/

3. Modeling portfolio returns and risk. Available at: https://pidru4niki.com/15660721/investuvannya/modelyuvannya_dohidnosti_riziku_portfely

4. Yu Pengfei & Yan Xuesong. (2020). Stock price prediction based on deep neural networks. *Neural Computing and Applications.* 32. 10.1007/s00521-019-04212-x. Available at: https://www.researchgate.net/publication/332488706_Stock_price_prediction_based_on_deep_neural_networks

5. Lazarenko I., Krykun Y. Potrfolio management with time series analysis methods. Available at: https://ev.fmm.kpi.ua/article/view/309267/300787

6. Neural Network Layers: All You Need Is Inside Comprehensive Overview. Available at: https://hackernoon.com/neural-network-layers-all-you-need-is-an-inside-comprehensive-overview

7. Rectifier (neural networks). Available at: https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

8. Wenxiang Li, K. L. Eddie Law. Deep Learning Models for Time Series Forecasting: A Review. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10583885

Стаття надійшла: 26.08.2025
Стаття прийнята: 14.09.2025
Стаття опублікована: 09.10.2025