Proceedings of CONF-BPS 2026 Symposium: GenAI, Labour Markets, and the Economics of Human and Financial Capital
DOI: 10.54254/2754-1169/2026.LD31285

A Review of Markowitz Portfolio Theory Based on the Application of Python in Financial Risk Management

Tongyu Wu

Shanghai Lixin University of Accounting and Finance, Shanghai, China
3466561898@qq.com

Abstract. The financial market situation is unpredictable. Markowitz's investment theory is the cornerstone of the financial investment field, aiming to diversify risks and improve returns through precise parameter estimation. Entering the 21st century, with the advent of the big data era, Python can leverage its powerful computational capabilities to utilize this model more efficiently and accurately, helping the financial market operate stably. This review synthesizes findings from empirical studies that use Python tools to select real market data, analyze this data to obtain the optimal investment portfolio with the highest Sharpe ratio and the smallest variance, and conduct a comparative analysis of their expected returns, standard deviations, ultimately presenting the efficient frontier of the investment portfolio. The analysis concludes that using Python to analyze the Markowitz investment theory model can fully demonstrate a more intuitive, scientific, and precise quantitative financial research approach, assisting leaders in making rational investment decisions and enhancing risk management capabilities.

Keywords: Markowitz model, Python, Sharpe ratio, efficient frontier, optimal portfolio

1. Introduction

The goal of a mature financial risk management system is to balance and effectively manage risks through scientific asset allocation in dynamic market changes. In 1952, Harry Markowitz proposed the mean-variance model in his paper "Portfolio Selection," marking the birth of modern portfolio theory. This theory quantifies the expected return and risk of a portfolio as the mean and variance of asset returns, respectively, and by constructing an efficient frontier, it guides investors to seek either maximum returns for a given level of risk or minimum risk for a target return [1]. With the rapid development of China's financial markets, quantitative investment driven by big data has become a core method for asset allocation and risk management. Although the Markowitz portfolio model is theoretically nearly perfect, it faces challenges in practical applications due to complex calculations. First, the calculation of portfolio variance relies on the covariance matrix between pairs of assets, whose dimensionality grows quadratically with the number of assets. Simultaneously, solving for the efficient frontier is also a complex quadratic programming problem [2]. In traditional financial data processing, manual calculations or conventional data processing software like Excel are often used, which lack flexibility, are inefficient, and struggle to handle large-scale assets and complex

DOI: 10.54254/2754-1169/2026.LD31285

investment constraints. This significantly limits the application of Markowitz theory in today's financial markets.

With the advent of the big data era in the 21st century, Python, as an efficient open-source programming language, has gradually become an indispensable tool in the field of quantitative investment due to its advantages in data acquisition, computation, modeling, and visualization. As Zhu elaborated in "Investment Theory and Its Python Applications," Python's rich libraries (such as Pandas and Numpy) can efficiently process vast financial time series data [3]; libraries like Scipy provide powerful optimization algorithms for calculating portfolio weights; and tools like Matplotlib can visually display the efficient frontier and risk-return distributions. Additionally, Python's data processing capabilities are widely applied in derivative pricing, machine learning, and risk value (VaR) among other cutting-edge areas, providing solid technical support for the application of Python in the Markowitz model [4].

Currently, scholars are actively exploring developments in this field. For example, Sun conducted an empirical study on 20 A-share stocks using Python, clearly demonstrating the entire process from data acquisition (using the tushare library), covariance calculation, Monte Carlo simulation, to ultimately solving for the portfolio with the maximum Sharpe ratio and minimum variance. This study confirmed the significant value of Python in the practical application of Markowitz theory [5].

In existing research, studies have attempted to explore the applicability of Python in the Chinese market [6], but there remains a lack of empirical analysis combining modern technological methods. Most studies still rely on traditional financial data processing software like Excel, failing to demonstrate and evaluate the effectiveness of Python in data extraction, computation, and visualization, let alone preemptively addressing potential risks such as computational inaccuracies or high usage costs, especially under specific constraints in China's financial markets (e.g., short-selling prohibitions). How to systematically leverage Python to bridge the gap from theory to practice requires further exploration.

Based on the above content, this review article first elaborates on the core framework of Markowitz theory and its implications for risk management. It then describes the key technical processes, empirical approaches, and extended applications of implementing this model with Python in the field of financial risk management. Finally, by analyzing the progress, achievements, and shortcomings of existing research, it aims to provide more valuable insights for the future integration of digital technology and financial risk management.

2. Literature survey

This paper systematically reviews the research on the application of Python in Markowitz's portfolio theory. Existing literature shows that Python, with its robust open-source ecosystem, has evolved from a basic implementation tool to a core driver of theoretical innovation. Related studies primarily focus on dynamic risk modeling, multi-factor return prediction, intelligent optimization algorithms, and the integration of machine learning. Through empirical analysis, scholars have validated the significant effects of models such as GARCH and methods like HRP in enhancing portfolio robustness, injecting new vitality into traditional MPT. Future research should further explore cutting-edge directions such as intelligent decision-making and alternative data integration.

Figure 1 shows the trend in the number of publications with the keywords "Python" and "Markowitz portfolio" from 2015 to 2023. The number of related research papers grew rapidly from fewer than 200 in 2015 to over 1,800 in 2023, with an average annual growth rate of 35%. Particularly after 2020, with the maturation of open-source libraries such as PyPortfolioOpt, the number of publications exhibited an accelerated growth trend. This significant trend fully

DOI: 10.54254/2754-1169/2026.LD31285

demonstrates that research on the Markowitz model based on Python is becoming an important direction in the fields of financial engineering and quantitative investment, with its application prospects attracting considerable attention from academia.

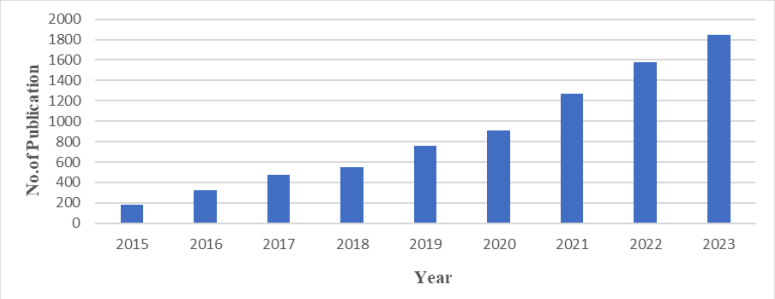

Figure 1. The number of papers searched using “Python” and “Markowitz portfolio” per year

3. Advanced applications of Python in Markowitz portfolio theory classification

3.1. Implementing risk modeling from static covariance to dynamic volatility

Traditional Markowitz portfolio theory is the cornerstone of modern finance, but one of its core assumptions—that asset returns follow a static, unchanging joint distribution—is severely inconsistent with the high-frequency, time-varying characteristics of financial markets. This theory relies on the historical sample covariance matrix to estimate future risk, which ignores key phenomena prevalent in financial time series, such as volatility clustering and time-varying correlations. The groundbreaking research first systematically proposed the Autoregressive Conditional Heteroskedasticity (ARCH) model, laying the econometric foundation for characterizing and modeling such time-varying volatility characteristics, thereby directly promoting a fundamental shift in risk modeling from static to dynamic [7].

To address the static shortcomings of the sample covariance, academia and industry widely adopt GARCH family models (such as GARCH, DCC-GARCH) to dynamically estimate the conditional covariance and embed it into the mean-variance optimization framework. Wu pointed out that dynamic risk models are not only a theoretical improvement but also a necessary tool for dealing with complex phenomena in real markets, such as the "volatility smile" and sudden changes in correlation [8]. This evolution marks a cognitive leap in risk modeling from a "static world" to a "dynamic process."

Zheng and Liu utilized Python's arch library to construct a GARCH(1,1) model, dynamically estimating asset volatility and correlation coefficients, and combined it with PyPortfolioOpt for weight optimization [9]. The results showed that in out-of-sample testing, the dynamic covariance model significantly reduced portfolio volatility and maximum drawdown compared to the static model, verifying the effectiveness of the dynamic model. To further enhance the model's adaptability to different market environments, Li and Wang combined the five-factor asset pricing model proposed by Fama and French with DCC-GARCH to construct a conditional covariance matrix [10,11]. This integration allows the portfolio's risk estimation to not only reflect the time-varying volatility characteristics of the market but also dynamically capture the exposure and changes of key risk factors such as size, value, profitability, and investment style, thereby constructing a more adaptive portfolio.

When the number of assets is large, both static sample covariance and dynamic GARCH models face the "curse of dimensionality" problem - too many parameters to estimate, leading to matrix

DOI: 10.54254/2754-1169/2026.LD31285

instability and amplification of estimation errors. This is particularly prominent when using big data for high-frequency or cross-market asset allocation. To address this, Ledoit and Wolf proposed the classic shrinkage estimation method, which achieves an optimal balance between bias and variance by shrinking the high-dimensional, unstable sample covariance matrix towards a structured, stable target matrix (such as a constant correlation matrix or a single-factor model matrix) [12]. This method significantly improves the condition number of the high-dimensional covariance matrix, greatly enhancing the robustness and practicality of portfolio optimization results under the Markowitz framework, making it one of the indispensable standard tools for handling large-scale asset allocation problems.

With the development of computing technology, more complex methods have been introduced into risk modeling. Machine learning algorithms (such as random forests, neural networks) can capture nonlinear, non-stationary fluctuation patterns from massive amounts of structured and unstructured data, providing new tools for predicting conditional volatility. At the level of optimization solving, traditional quadratic programming algorithms may encounter difficulties when facing complex constraints (such as integer constraints, transaction costs, Value at Risk constraints). The convex optimization theory provides a powerful unified mathematical framework and efficient solving algorithms (such as interior-point methods) for this, enabling large-scale portfolio optimization problems containing complex constraints and more precise risk models to be solved reliably and efficiently, greatly expanding the practical boundaries of portfolio theory [13].

3.2. Multi-factor models and macroeconomic variables for return prediction

To overcome the "estimation error" problem of historical mean returns, Python supports incorporating multi-factor models (e.g., Fama-French three-factor, five-factor models) and macroeconomic variables into the expected return estimation system.

Chen used Python's statsmodels library to estimate factor returns and called PyPortfolioOpt for mean-variance optimization [14]. The study found that factor-adjusted expected returns better captured cross-sectional return differences, and the optimization results were more economically interpretable. Zhang introduced macroeconomic variables such as inflation and interest rates as return prediction factors, constructing dynamic expected returns, further improving the model’s robustness during economic cycles [15].

The deepening of theory and the evolution of optimization tools are key to this field. The Bayesian mixed model provides a classic framework for integrating investors' subjective views with market equilibrium returns (which can be derived from multi-factor models), greatly improving the defect of mean-variance optimization being overly sensitive to input is a cornerstone methodology for building robust expected return systems [16]. At the practical level, as it transforms theoretical predictions into practically executable strategies, quantifies the erosion of optimization results by factors such as turnover rate and bid-ask spread, and ensures that the model moves from "theoretically superior" to "practically profitable" [17]. Finally, as factors and macroeconomic variables increase, traditional optimization methods may face dimensionality challenges. de Prado systematically explores how to apply machine learning methods for dimensionality reduction, nonlinear relationship modeling, and preventing overfitting, providing cutting-edge technical perspectives and toolkits for handling high-dimensional, complex factor-return relationships [18].

DOI: 10.54254/2754-1169/2026.LD31285

3.3. Extending optimization algorithms from quadratic programming to intelligent optimization

Although Markowitz's mean-variance model is essentially a quadratic programming problem, laying the cornerstone for modern portfolio theory [19], real-world investing often faces complex scenarios, such as integer constraints (e.g., minimum trading lots), dynamic transaction costs, liquidity restrictions, and tail risk constraints. These complex factors, together with the richer sources of risk and return revealed by multi-factor pricing models [11], require that the optimization framework must go beyond the traditional convex optimization category to handle non-smooth, non-convex, and even NP-hard portfolio optimization problems.

The Python ecosystem provides powerful and flexible support for this, forming a complete toolchain ranging from traditional optimization to intelligent algorithms. On one hand, libraries such as CVXPY and SciPy can efficiently handle convex optimization problems with linear and quadratic constraints; on the other hand, intelligent optimization algorithms represented by genetic algorithms, particle swarm optimization, and simulated annealing can effectively explore non-convex, discrete, and high-dimensional solution spaces. For example, in their research, Liu and Zhang utilized Python's DEAP library to construct a multi-objective genetic algorithm, which not only simultaneously optimized the three objectives of return, risk, and liquidity but also could naturally handle the cardinality constraints of asset weights (i.e., the upper limit on the number of invested assets), which is difficult to achieve directly with traditional quadratic programming [20].

Such practices highlight Python's core position in financial modeling and computation. As Hilpisch explained, Python has become a key tool for realizing data-driven finance and connecting classical theories with cutting-edge algorithms [21]. This connection is reflected not only in algorithm calls but also in the integration capability of the entire process: researchers can use pandas and NumPy for data cleaning and factor calculation, use scikit-learn to build machine learning models for predicting returns or risks, and finally complete the solution of portfolio weights through CVXPY or intelligent optimization libraries [22]. Zhang discussed improving the Markowitz model based on the time-varying nature of mean and variance; the solution of such dynamic models often also relies on more complex numerical optimization techniques [23].

3.4. From mean-variance model to Hierarchical Risk Parity (HRP)

Starting from Markowitz's mean-variance model (MPT) [24], it seeks a balance between risk and return through mathematical optimization, laying the cornerstone for quantitative asset allocation. However, MPT is extremely sensitive to input parameters (expected returns and covariance matrix), and in high-dimensional, non-normal, or strongly correlated market environments, its estimation errors are amplified, leading to concentrated and unstable portfolio weights, especially performing poorly during periods of market stress (the "error maximization" problem).

To overcome these limitations, research and practice began to seek allocation methods that rely less on model assumptions and are more robust. This shift is deeply influenced by the trends in data science and machine learning [25]. Researchers no longer solely rely on point estimates from historical data but instead attempt to directly mine the intrinsic structure among assets from the data. For example, by analyzing large-scale stock data, it can be discovered that the correlations among assets are not random but exhibit certain hierarchical clustering characteristics [26], meaning factors such as industry and style cause assets to naturally cluster in terms of risk dimensions.

Against this background, Hierarchical Risk Parity (HRP) emerged, representing an important direction of evolution. The core innovation of HRP lies in its complete abandonment of estimating

DOI: 10.54254/2754-1169/2026.LD31285

expected returns (focusing only on risk) and its clever utilization of the hierarchical clustering structure among assets to construct the portfolio [27]. Its algorithmic process (typically efficiently implemented via Python can be summarized as: 1) Calculate a distance matrix based on asset returns to measure "dissimilarity"; 2) Apply a hierarchical clustering algorithm to organize assets into a dendrogram (cluster tree); 3) Perform recursive bisection and risk budget allocation along the cluster tree (often using simple risk parity, i.e., equal risk contribution) to ultimately determine the weights of each asset [28].

The evolution from the mean-variance model (MPT) to Hierarchical Risk Parity (HRP) is a typical example of the development of asset allocation theory from parameter-sensitive optimization towards structure-driven robustness. HRP inherits MPT's pursuit of risk diversification but, by introducing clustering concepts from machine learning, effectively mitigates the sensitivity to input parameters and more naturally utilizes the hierarchical correlation structures present in financial markets. Consequently, in practice, especially in complex and high-dimensional investment environments, it provides a more stable and reliable asset allocation solution compared to traditional MPT [27]. This evolutionary path profoundly reflects the deep integration of modern financial engineering and data science technologies.

3.5. Visualization and interactive analysis: enhancing decision support capabilities

The core mathematical models of portfolio optimization (such as the Markowitz mean-variance model) provide a theoretical foundation for decision-making, while modern Python libraries (such as Matplotlib, Plotly, and Dash) transform these theories into intuitive visual outputs like the efficient frontier, weight distribution, and sensitivity analysis, and support the construction of interactive graphical user interfaces, greatly enhancing the model's operability and transparency.

This practical path of "theory visualization" is clearly reflected in recent research. Hu precisely utilized the Dash framework to develop a complete web application [29]. This system allows users to upload data, dynamically set risk preference parameters, and view optimization results and graphical feedback in real-time, significantly lowering the usage threshold of complex models, making it a typical example of interactive decision support. The underlying data processing capabilities of such systems rely on the support of efficient data structures like Pandas, ensuring computational fluency from raw data to visual results [30].

The value of visualization lies not only in presenting results but also in deep analysis and exploration. Tang pointed out that information visualization technology can help investors quickly identify patterns and anomalies from massive, multi-dimensional financial data [31]. This complements the rich visualization code provided by many open-source projects (such as the official PyPortfolioOpt examples), collectively helping investors intuitively understand the characteristics and risk sources of a portfolio.

Furthermore, visualization and interactive tools have also revolutionized traditional teaching methods and model-building approaches. Yang specifically mentioned the fundamental role of Excel's visualization and simulation functions in the teaching and application of investment models [32]. With the evolution of the tool ecosystem, contemporary analytical platforms have expanded from Excel to more powerful programming environments. Just as Chen emphasized in econometric applications the importance of choosing appropriate software tools (such as EViews, SAS), in the field of portfolio analysis, the Python-based interactive visualization ecosystem (Dash/Streamlit) has become a key bridge connecting cutting-edge theory, econometric methods, and end-user decision-making, marking a development of decision support systems towards a more intelligent and inclusive direction [33].

DOI: 10.54254/2754-1169/2026.LD31285

4. Future directions

4.1. Intelligence and adaptive learning

Future portfolio optimization will place greater emphasis on the intelligence level of models. Reinforcement learning-based dynamic asset allocation methods will become a research hotspot, with systems capable of automatically adjusting risk preferences and investment objectives based on market conditions. The integration of deep reinforcement learning frameworks such as TensorFlow and PyTorch with traditional optimization libraries will enable portfolios to possess continuous learning and self-optimization capabilities.

4.2. Big data and alternative data integration

With the increasing richness of alternative data sources, the Python ecosystem will play a larger role in processing unstructured data. New data sources such as social media sentiment, satellite imagery, and supply chain data will be incorporated into expected return and risk assessment systems through natural language processing (NLP) and computer vision technologies, providing more comprehensive inputs for traditional mean-variance models.

4.3. Real-time risk monitoring and dynamic rebalancing

Leveraging Python's high-performance computing libraries (e.g., Numba, Dask), future portfolio management systems will achieve near real-time risk monitoring and dynamic rebalancing. Combined with stream data processing technologies (e.g., Apache Kafka), systems will be able to respond to market changes at minute or even second levels, significantly enhancing the foresight and timeliness of risk management.

4.4. Multi-objective collaborative optimization

Future portfolio optimization will transcend the traditional two-dimensional risk-return framework and evolve toward multi-objective collaborative optimization. Python's multi-objective optimization libraries (e.g., PyMOO) will be widely applied to complex decision-making scenarios that simultaneously consider multiple dimensions such as returns, risks, liquidity, ESG metrics, and social impact, meeting the increasingly diversified investment needs of modern investors.

4.5. Application of explainable AI (XAI) in portfolio decision-making

As the complexity of machine learning models increases, model interpretability becomes increasingly important. Python-based explainable AI tools such as SHAP and LIME will be integrated into portfolio analysis workflows, helping investors understand the decision logic of complex models (e.g., neural networks, ensemble learning) and enhancing trust and acceptance of portfolio allocation results.

4.6. Cloud collaboration and open-source ecosystem development

The Python open-source ecosystem will continue to democratize portfolio management technologies. Cloud platform-based (e.g., AWS, Azure) collaborative research environments will facilitate knowledge sharing between academia and industry, while the continuous improvement of

DOI: 10.54254/2754-1169/2026.LD31285

open-source projects such as PyPortfolioOpt and Zipline will provide more professional and user-friendly analytical tools for various investors.

4.7. Cross-market linkages and global asset allocation

Against the backdrop of global integration, Python's capabilities in geospatial analysis and cross-market correlation modeling will be fully utilized. Combining global macroeconomic data to build multi-country, multi-asset class allocation models that consider exchange rate risks and geopolitical factors will become an important research direction for institutional investors.

4.8. Regulatory technology (regtech) integration

As regulatory requirements become increasingly stringent, future Python portfolio tools will deeply integrate compliance checking functionalities. Through smart contracts and rule engines, real-time compliance monitoring of portfolios will be achieved, automatically detecting violations of investment restrictions and excessive risk exposures, significantly improving compliance management efficiency.

5. Conclusion

This paper systematically elaborates on the innovative applications and development prospects of Python in Markowitz's portfolio theory. Research shows that, leveraging its robust open-source ecosystem, Python has evolved from a basic implementation tool into a core driver of theoretical innovation. At the application level, Python has achieved significant breakthroughs by integrating dynamic risk models such as GARCH, transitioning from static covariance to time-varying volatility; significantly improving the accuracy of return predictions through multi-factor models and machine learning algorithms; and effectively solving portfolio optimization problems under complex constraints using intelligent optimization techniques. Particularly noteworthy is the implementation of innovative methods like hierarchical risk parity and powerful visualization support, which have transformed Markowitz's theory from an ideal static model into a practical risk management framework adaptable to dynamic markets. Looking ahead, Python's applications in this field will exhibit four major development trends: intelligent decision-making based on deep reinforcement learning, big data analytics incorporating alternative data, real-time risk control supporting second-level responses, and collaborative optimization incorporating multi-dimensional objectives such as ESG. These developments will propel portfolio management toward smarter, more comprehensive, and more efficient directions, providing solid support for building a new-generation financial risk management system.

References

[1] Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7 (1), 77-91.
[2] Zhang, Z. Q. (2015). Research on Markowitz portfolio model with changes in mean and variance. Statistics & Decision, 12 , 152-155.
[3] Zhu, S. Q. (2019). Investment and its Python application . Tsinghua University Press.
[4] Li, Y., & Yu, L. X. (2013). Theoretical analysis of optimal securities portfolio based on Markowitz theory. Financial Theory and Practice, 35 (5), 78-81.
[5] Sun, L. B. (2022). An empirical study on Markowitz portfolio theory based on Python. Computer Engineering and Applications, 58 (18), 245-251.

DOI: 10.54254/2754-1169/2026.LD31285

[6] Liu, X. H. (2018). Portfolio construction based on Markowitz theory. Journal of Commercial Economics, 8 (9), 167-169.
[7] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica, 50 (4), 987-1007.
[8] Wu, C. F. (2020). Research on financial engineering . Science Press.
[9] Zheng, X. H., & Liu, Q. (2020). Research on portfolio optimization based on GARCH-mean-variance model. Journal of Systems Engineering, 35 (3), 389-400.
[10] Li, M., & Wang, F. (2021). An empirical study on the integration of multi-factor GARCH and Markowitz model. Journal of Applied Statistics and Management, 40 (4), 698-710.
[11] Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. Journal of Financial Economics, 116 (1), 1-22.
[12] Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. Journal of Multivariate Analysis, 88 (2), 365-411.
[13] Boyd, S., & Vandenberghe, L. (2004). Convex optimization . Cambridge University Press.
[14] Chen, J. (2019). A-share return prediction and portfolio optimization based on the Fama-French five-factor model. Investment Research, 38 (6), 123-138.
[15] Zhang, W. (2022). Research on Markowitz model enhanced by introducing macro factors. Journal of Financial Economics Research, 37 (2), 89-102.
[16] Black, F., & Litterman, R. (1992). Global portfolio optimization. Financial Analysts Journal, 48 (5), 28-43.
[17] Wang, X. D. (2020). Portfolio rebalancing model considering transaction costs and its Python implementation. Operations Research and Management Science, 29 (7), 18-25.
[18] de Prado, M. L. (2016). Advances in financial machine learning . John Wiley & Sons.
[19] Liu, K. D. (2018). Portfolio construction based on Markowitz theory. Modern Business, 517 (36), 186-187.
[20] Liu, R., & Zhang, J. H. (2021). Multi-objective portfolio optimization based on genetic algorithm. Control and Decision, 36 (8), 1917-1924.
[21] Hilpisch, Y. (2018). Python for finance: Mastering data-driven finance (2nd ed.). O'Reilly Media.
[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12 , 2825-2830.
[23] Zhang, H. Q. (2015). Research on Markowitz portfolio model with changes in mean and variance [Master's thesis]. Harbin Institute of Technology, Harbin.
[24] Zhang, J. Q. (2019). Financial risk management (4th ed.). Fudan University Press.
[25] Fletcher, S., & Gardner, J. (2019). Machine learning in finance: From theory to practice . Springer.
[26] Ye, F., Liang, W. L. W., & Zhang, D. Y. (2021). Mining of stock correlation structure characteristics and investment strategy research based on big data analysis and Python. Data Analysis and Knowledge Discovery, 5 (3), 112-120.
[27] Zhao, L. (2022). A comparative study of HRP and traditional MPT. Investment and Entrepreneurship, 15 , 134-136.
[28] Li, S. (2017). Implementation of financial applications based on Python scientific computing packages [Master's thesis]. Jiangxi University of Finance and Economics, Nanchang.
[29] Hu, T. (2021). Design and implementation of a portfolio optimization system based on Dash. Computer Applications and Software, 38 (11), 234-239+271.
[30] McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & K. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 51-56). SciPy.
[31] Tang, Q. N., Chen, Z. J., Tu, T. Y., & Wang, R. (2021). Research on stock data information visualization in the big data era. Computer Knowledge and Technology, 17 (8), 201-203.
[32] Yang, Z. (2018). Development and model construction of portfolio theory: Also on the application of Excel in investment models. Economic Research Guide, 386 (36), 56-58.
[33] Chen, D. T. (2012). Applied econometrics: Examples with EViews and SAS . Peking University Press.