DIRECTORATE OF ONLINE EDUCATION
(A unit of Manipal Academy of Higher Education, Manipal)

Comparative Analysis of ML Models Used in Portfolio Optimization

PROJECT REPORT SUBMITTED TO

MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

in partial fulfilment for the award of the degree

of

MASTER OF SCIENCE

Data Science

Submitted by:

Prajval P

Reg. No: 22154090137

Under the guidance of

University Project Mentor

Dr. Tarushree Bari

August, 2024

Contents

1. Introduction

Portfolio optimization is a crucial concept in finance, focusing on the selection of assets to maximize returns for a given level of risk or, conversely, to minimize risk for a given level of expected returns. The significance of portfolio optimization lies in its ability to help investors make informed decisions about asset allocation, balancing risk and return according to their financial goals and risk tolerance.

1.1 Importance of Portfolio Optimization in Finance

1. **Risk Management**: Portfolio optimization allows investors to manage and control the risk associated with their investments. By diversifying assets and selecting those with varying degrees of correlation, an optimized portfolio can reduce overall risk while maintaining the potential for returns.
2. **Maximizing Returns**: The primary goal of portfolio optimization is to achieve the highest possible return for a given level of risk. Through careful asset selection and weighting, investors can enhance their portfolio's performance.
3. **Efficient Resource Allocation**: Portfolio optimization helps in allocating resources efficiently by identifying the best combination of assets that align with an investor's risk-return profile. This leads to more effective use of capital and improved financial outcomes.
4. **Adapting to Market Conditions**: Markets are dynamic, and portfolio optimization allows for adjustments based on changing market conditions, economic factors, and individual asset performance. This adaptability is key to maintaining an effective investment strategy over time.
5. **Personalization**: Each investor has unique financial goals and risk tolerances. Portfolio optimization enables the creation of customized portfolios that align with individual investor needs, rather than a one-size-fits-all approach.

1.2 Role of Machine Learning Models in Portfolio Optimization

Machine learning (ML) models have become increasingly important in portfolio optimization due to their ability to handle large datasets, uncover complex patterns, and improve predictive accuracy. Here's how ML models contribute to portfolio optimization:

1. **Enhanced Predictive Analytics**: ML models can analyze vast amounts of historical and real-time data to predict asset returns, volatility, and other financial metrics more accurately than traditional methods. This helps in making more informed investment decisions.
2. **Dynamic Portfolio Adjustment**: ML algorithms can continuously learn from new data and adjust portfolios dynamically, ensuring that they remain optimized as market conditions change. This adaptive learning capability is a significant advantage in volatile markets.

3. **Non-Linear Relationships**: Traditional portfolio optimization methods often rely on linear assumptions (e.g., mean-variance optimization). ML models can capture non-linear relationships between assets, leading to better diversification and risk management strategies.
4. **Risk Assessment**: ML models can be used to assess and predict risks more effectively. For example, they can identify potential market downturns, detect anomalies, and predict tail risks, enabling investors to take proactive measures.
5. **Algorithmic Trading**: In algorithmic trading, ML models optimize portfolios by making split-second decisions on asset allocation based on real-time data. This enhances the efficiency and effectiveness of trading strategies.
6. **Personalized Portfolio Solutions**: ML can create highly personalized investment portfolios by analyzing individual investor behavior, preferences, and goals. This level of personalization can lead to more effective and satisfying investment experiences.
7. **Scenario Analysis and Stress Testing**: ML models can simulate various economic scenarios and perform stress tests on portfolios. This helps in understanding how portfolios might perform under different market conditions and prepares investors for potential risks.

The objective of this project is to perform a comparative analysis of Hierarchical Risk Parity (HRP) and Graphics Neural Networks (GNN) models for portfolio optimization.

2. Literature Review

Portfolio optimization has evolved significantly, starting from traditional methods like Markowitz Mean-Variance Optimization (MVO) to more advanced techniques like Hierarchical Risk Parity (HRP) and Graph Neural Networks (GNNs). Each method has its strengths and limitations, with newer techniques addressing some of the challenges posed by traditional approaches.

2.1 Traditional Methods: Markowitz Mean-Variance Optimization (MVO)

Markowitz Mean-Variance Optimization is one of the foundational methods in portfolio optimization, introduced by Harry Markowitz in 1952. It is based on the idea of balancing risk and return by selecting a mix of assets that minimizes the portfolio's variance (risk) for a given expected return or maximizes return for a given level of risk.

Key Concepts:

1. **Expected Return**: The weighted average of the expected returns of the assets in the portfolio.
2. **Risk (Variance/Standard Deviation)**: The variance (or standard deviation) of the portfolio's return, representing the risk.
3. **Covariance**: Measures how two assets move together, which is crucial for determining the overall portfolio risk.

Strengths:

- **Foundational Framework**: MVO provides a clear, mathematically sound framework for thinking about the trade-off between risk and return.
- **Diversification**: Encourages diversification by considering the correlations between assets, aiming to reduce unsystematic risk.

Limitations:

- **Assumptions**: MVO assumes that returns are normally distributed and that investors have a quadratic utility function, which may not hold true in real-world scenarios.
- **Sensitivity to Input Estimates**: The optimization is highly sensitive to estimates of expected returns, variances, and covariances, which are difficult to predict accurately.
- **Overemphasis on Risk**: MVO tends to favor portfolios with lower volatility, which may not always align with investor preferences or market conditions.

2.2 Advanced Techniques

To address the limitations of traditional methods like MVO, several advanced techniques have been developed, including Hierarchical Risk Parity (HRP) and Graph Neural Networks (GNNs).

1. Hierarchical Risk Parity (HRP)

Hierarchical Risk Parity (HRP) is a more recent approach to portfolio optimization introduced by Marcos López de Prado in 2016. HRP addresses some of the shortcomings of traditional methods, particularly in handling the instability and sensitivity of covariance matrices used in MVO.

Key Concepts:

- **Clustering**: HRP uses hierarchical clustering to group assets into clusters based on their correlation. This clustering helps in identifying relationships between assets that are not immediately apparent.
- **Tree-Based Allocation**: After clustering, HRP uses a tree-based approach to allocate risk across the portfolio, ensuring that the risk is distributed more evenly.

Strengths:

- **Robustness**: HRP is more robust to estimation errors, particularly in the covariance matrix, which makes it less sensitive to outliers and noise in the data.
- **Improved Diversification**: By focusing on the hierarchical structure of asset correlations, HRP can achieve better diversification compared to MVO.
- **Scalability**: HRP scales well with the number of assets, making it suitable for large portfolios.

Limitations:

- **Complexity**: HRP is more complex than MVO, requiring more advanced understanding and computational resources.
- **Interpretability**: The hierarchical clustering process can be less intuitive compared to the straightforward optimization framework of MVO.

2. Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) represent the frontier of machine learning applied to portfolio optimization. GNNs are a type of neural network that operates on graph structures, which are particularly useful in capturing the relationships and interactions between different assets in a portfolio.

Key Concepts:

- **Graph Representation**: Assets are represented as nodes in a graph, with edges representing the relationships (e.g., correlations) between them.
- **Learning Asset Interactions**: GNNs can learn complex, non-linear interactions between assets, going beyond the linear assumptions of traditional methods.

- **End-to-End Learning:** GNNs can be trained end-to-end, directly optimizing for the portfolio's risk-return trade-off, taking into account the graph structure.

Strengths:

- **Capturing Complex Dependencies**: GNNs excel at capturing the complex, non-linear dependencies between assets, which traditional methods might miss.
- **Flexibility**: GNNs can be adapted to various types of data and can incorporate additional features (e.g., macroeconomic indicators, sentiment analysis) into the optimization process.
- **State-of-the-Art Performance**: GNNs have shown superior performance in back testing scenarios, particularly in environments with complex, evolving asset relationships.

Limitations:

- **Data Requirements**: GNNs require large amounts of data and are computationally intensive, which might be a barrier for some applications.
- **Black-Box Nature**: Like many machine learning models, GNNs can be difficult to interpret, making it hard to understand the rationale behind certain portfolio decisions.

3. Methodology

3.1 Data Collection

The dataset consists of 2,955 entries (rows) and 902 features (columns). The data appears to contain stock market information for various companies, including price data, trading volumes, and target returns. Here is an overview of the dataset and its features:

Dataset Overview:

- **Number of Rows:** 2,955
- **Number of Columns:** 902

Key Features:

1. **Date**: The timestamp for each entry, indicating the date on which the stock data was recorded.
2. **Stock Features**: For each stock, multiple features are provided, which include:
    - **Open**: The opening price of the stock on the given date.
    - **High**: The highest price reached by the stock on that date.
    - **Low**: The lowest price reached by the stock on that date.
    - **Close**: The closing price of the stock on that date.
    - **Volume**: The number of shares traded on that date.
    - **Dividends**: Any dividends paid out on that date.
    - **Stock Splits**: Information on any stock splits that occurred.
3. **Target Returns**: Each stock also has a corresponding target return, which likely represents the predicted or actual return for that stock over a certain period. The target returns are prefixed with the stock's ticker symbol followed by _target_returns.
4. Domains and Representative Companies:
    - **Technology**: Tata Consultancy Services (TCS), Infosys, Wipro, HCL Technologies, HCL Technologies, Mphasis, Coforge and Persistent Systems.
    - **Finance**: HDFC Bank, ICICI Bank, State Bank of India, Kotak Mahindra Bank, Axis Bank, Bajaj Finance, IndusInd Bank, IDFC First Bank and Bank of Baroda.
    - **Energy**: Reliance Industries, Oil and Natural Gas Corporation, Indian Oil Corporation, NTPC, Power Grid Corporation, Tata Power, Adani Green Energy, GAIL India, Bharat Petroleum and Hindustan Petroleum.

- **Consumer Goods:** Hindustan Unilever, ITC Ltd., Nestle India, Dabur India, Brittania Industries, Marico, Colgate-Palmolive India, Godrej Consumer Products, Emami and Tata Consumer Products.
- **Healthcare:** Dr. Reddy's Laboratories, Sun Pharmaceutical, Cipla, Lupin, Biocon, Aurobindo Pharma, Divi's Laboratories, Apollo Hospitals, Fortis Healthcare and Glenmark Pharmaceuticals.
- **Automobile:** Maruti Suzuki, Tata Motors, Mahindra & Mahindra, Bajaj Auto, Hero MotoCorp, Ashok Leyland, TCS Motor Company, Eicher Motors, Motherson Sumi Systems and Bosh Ltd.
- **Metals and Mining:** Tata Steel, Hindalco Industries, JSW Steel, Vedanta Ltd., National Aluminium Company, COAL India, Steel Authority of India, NMDC Ltd., Jindal Steel and Power and Hindustan Zinc.
- **Real Estate:** DLF Ltd., Godrej Properties, Oberoi Realty, Phoenix Mills, Prestige Estates, Brigade Enterprises, Sobha Ltd., Indiabulls Real Estate, Purvankara and Mahindra Lifespace Developers.

3.2 Model Implementation

3.2.1 Hierarchical Risk Parity (HRP):

- The correlation matrix of the stock returns is computed to understand the relationships between different stocks. This matrix is fundamental for identifying clusters of stocks that move similarly.
- To perform clustering, the correlation matrix is transformed into a distance matrix using the Euclidean distance metric. This conversion quantifies the dissimilarity between each pair of stocks based on their return correlations.
- Hierarchical clustering: It is performed on the distance matrix to create a linkage matrix. This matrix captures the hierarchical relationships between stocks, which is essential for the HRP method.
- Portfolio Weights Calculation (Simplified HRP): In a complete HRP implementation, the hierarchical clustering results would be used to recursively allocate portfolio weights based on the tree structure. However, in this simplified version, equal weights are assigned to all assets as a placeholder.
- Calculating Portfolio Returns and Performance Metrics: The portfolio returns are calculated by taking the dot product of the stock returns and the portfolio weights. Subsequently, cumulative returns and the Sharpe ratio (a measure of risk-adjusted return) are computed.

The HRP model aims to construct diversified portfolios by leveraging the hierarchical structure of asset correlations. While the provided implementation includes the initial steps of clustering and a simplified weight allocation, a complete HRP approach would involve a recursive bisection process to optimally allocate weights based on the hierarchical tree. This method addresses some limitations of traditional portfolio optimization techniques, such as sensitivity to input estimates and assumptions about asset return distributions, by focusing on the inherent structure and relationships within the data.

3.2.2 Graphics Neural Networks (GNN)

The Graph Neural Network (GNN) architecture implemented in the code is designed to predict stock returns by leveraging the relationships between different assets, modelled as a fully connected graph. Below is a detailed description of the key components used in this architecture:

1. Graph Attention Networks (GATConv)

- **Purpose**: GATConv layers are a type of convolutional layer used in GNNs. They apply attention mechanisms to aggregate information from a node's neighbors, allowing the model to focus on the most relevant nodes during each convolutional step.
- **Implementation**:
    - The first GATConv layer (conv1) takes the number of input features as the input size and outputs 128 features per head. The heads=4 parameter indicates that multi-head attention is used, meaning that four separate attention mechanisms are applied in parallel, and their outputs are concatenated.
    - The second GATConv layer (conv2) further processes the output of the first layer, reducing the feature size to 64 per head, again using four heads.

2. Batch Normalization (BatchNorm)

- **Purpose**: Batch Normalization is applied after each GATConv layer to normalize the outputs. This helps in stabilizing and accelerating the training process by reducing internal covariate shifts.
- **Implementation**:
    - After each GATConv layer, a BatchNorm layer is applied to the output, normalizing it across the batch.

3. Fully Connected Layers

- **Purpose**: After the convolutional layers, the GNN uses fully connected (dense) layers to map the high-dimensional node features to the final asset returns.
- **Implementation**:
    - The first fully connected layer (fc1) reduces the dimensionality from 256 (output of the second GATConv) to 32.
    - The second fully connected layer (fc2) maps these 32 features to the number of assets, providing a prediction for the returns of each asset.

4. Dropout

- **Purpose**: Dropout is used as a regularization technique to prevent overfitting by randomly setting a fraction of the input units to zero during training.

- **Implementation**:
    - A dropout layer with a dropout rate of 0.3 is applied after each fully connected layer.

5. Custom Sharpe Ratio Loss Function (SharpeLoss)

- **Purpose**: The loss function is designed to maximize the Sharpe Ratio, a common metric in finance that measures the performance of an investment compared to a risk-free asset, after adjusting for its risk.
- **Implementation**:
    - The Sharpe ratio is calculated as the mean return divided by the standard deviation of returns. The loss function returns the negative Sharpe Ratio, which the optimizer will then minimize, effectively maximizing the Sharpe Ratio.

2.3 Evaluation Metrics

2.3.1 Cumulative Returns

Cumulative Returns measure the total return of an investment or portfolio over a specific period. It is the aggregate amount that an investment has gained or lost over time.

Equation:

Cumulative Returns = $\sum_{t=1}^{T} R_t$

Where:

- R is the return of the portfolio at time t.
- T is the total number of time periods.

In the code, Cumulative Returns are calculated by summing the predicted returns over time to assess the overall growth or decline in portfolio value.

2.3.2 Sharpe Ratio

The Sharpe Ratio is a measure of risk-adjusted return. It helps to understand how much excess return (over a risk-free rate) is generated per unit of risk taken by the portfolio.

Equation:

Sharpe Ratio = $\frac{E[R_p - R_f]}{\sigma_p}$

Where:

- R_p is the portfolio return.
- R_f is the risk-free rate of return (often assumed to be 0 in some models).
- σ_p is the standard deviation of the portfolio returns, representing the portfolio's risk.

In the code, the Sharpe Ratio is used to evaluate how well the portfolio compensates for the risk it takes, with a higher Sharpe Ratio indicating a more favorable risk-adjusted return.

2.3.3 Sortino Ratio

The Sortino Ratio is a variation of the Sharpe Ratio that focuses on downside risk rather than total volatility. It is particularly useful for assessing the risk of negative returns.

Equation:

$$\text{Sortino Ratio} = \frac{E[R_p - R_f]}{\sigma_d}$$

Where:

- $\mathrm{E}[R_p - R_f]$ is the expected excess return of the portfolio over the risk-free rate.
- σ_d is the downside deviation, which only considers the volatility of returns that fall below a certain threshold (e.g., 0 or a minimum acceptable return).

In the code, the Sortino Ratio is used to assess the portfolio's performance with a focus on avoiding downside risk.

2.3.4 Maximum Drawdown

Maximum Drawdown (MDD) is a measure of the largest drop in portfolio value from a peak to a trough, reflecting the worst case of loss during a specific period.

Equation:

$$\text{Maximum Drawdown} = \frac{\min(V_t) - \max(V_t)}{\max(V_t)}$$

Where:

- V_t is the portfolio value at time ttt.
- $\mathrm{Max}(V_t)$ is the highest value of the portfolio before the drawdown.
- $\mathrm{Min}(V_t)$ is the lowest portfolio value following the peak.

In the code, Maximum Drawdown is calculated to understand the potential worst-case scenario for the portfolio, providing insights into the portfolio's risk during market downturns.

4. Results

The performance metrics for the Hierarchical Risk Parity (HRP) and Graph Neural Network (GNN) models are evaluated using key financial indicators such as Cumulative Returns, Sharpe Ratio, Sortino Ratio, and Maximum Drawdown. These metrics help in comparing the risk-adjusted returns and overall performance of the two models.

Metric	HRP Model	GNN Model
Cumulative Returns	2.6226	Sum of cumulative returns over all periods
Sharpe Ratio	1.6888	0.8431
Sortino Ratio	0.1308	3.5325
Maximum Drawdown	3.1706	-0.3313

Sensitivity Analysis: Sharpe Ratios for Different Learning Rates and Layers in the GNN Model

The sensitivity analysis involves evaluating the Sharpe Ratio across different combinations of learning rates and the number of layers in the Graph Neural Network (GNN) model. The Sharpe Ratio is a key metric used to assess the risk-adjusted return of a portfolio, and in this context, it helps determine how the GNN model's architecture and hyperparameters influence its performance.

Heatmap of Sharpe Ratios

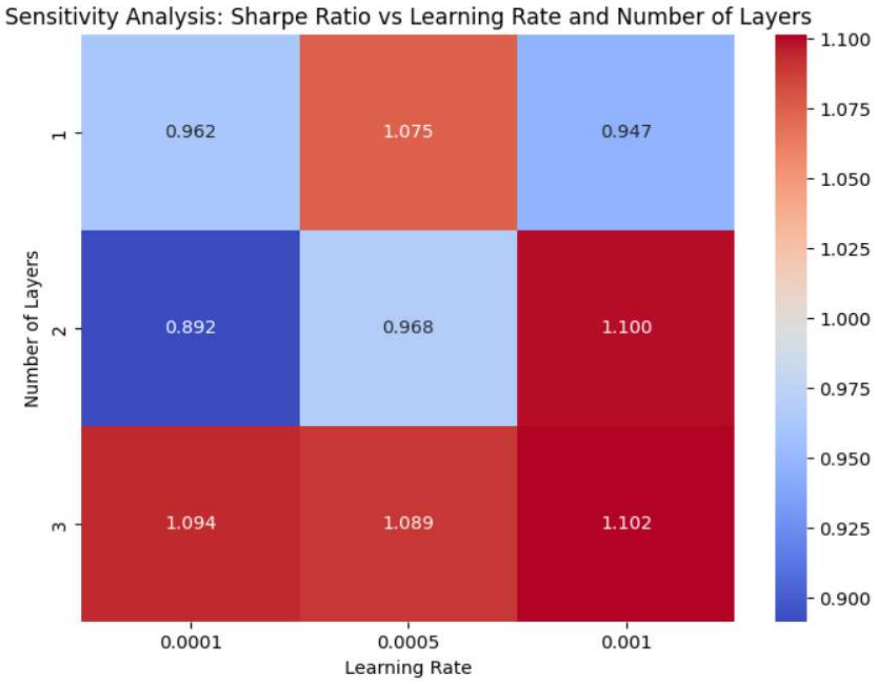

Figure: Heatmap of Sharpe Ratio vs Learning Rate and Number of Layers

The following table shows the Sharpe Ratios for various learning rates and the number of layers:

Findings from the Sensitivity Analysis

Learning Rate	Number of Layers	Sharpe Ratio (GNN)
0.0001	**1**	**0.9622**
0.0001	**2**	**0.8916**
0.0001	**3**	**1.0937**
0.0005	**1**	**1.0748**
0.0005	**2**	**0.9676**
0.0005	**3**	**1.0893**
0.0010	**1**	**0.9469**
0.0010	**2**	**1.0996**
0.0010	**3**	**1.1015**

1. **Learning Rate Impact:**
    - At a low learning rate of **0.0001**, the Sharpe Ratio is relatively low for fewer layers (1-2 layers) but increases significantly with 3 layers. This indicates that for this learning rate, a deeper model is necessary to achieve better performance.
    - A **0.0005** learning rate generally produces high Sharpe Ratios, particularly with 1 and 3 layers, making it an effective learning rate across different model depths.
    - At the highest learning rate of **0.0010**, the model's performance improves with more layers, achieving the highest Sharpe Ratio of **1.101547** with 3 layers.
2. **Impact of Number of Layers:**
    - With a **0.0001** learning rate, increasing the number of layers improves the Sharpe Ratio, with the highest value seen at 3 layers. This suggests that for lower learning rates, a deeper model can capture more complex relationships in the data.
    - For a **0.0005** learning rate, the Sharpe Ratio is high with both 1 and 3 layers, indicating that this learning rate is well-suited for models with varying complexity.
    - At **0.0010**, the Sharpe Ratio improves with more layers, showing that the model can handle a higher learning rate effectively when it has enough depth (3 layers).

Summary of Sensitivity Analysis

The sensitivity analysis reveals that both the learning rate and the number of layers are crucial in determining the performance of the GNN model:

- **Optimal Learning Rate:** The learning rate of **0.0005** is robust across different model depths, offering a good balance between training stability and performance.
- **Optimal Model Depth:** For the **0.0001** and **0.0010** learning rates, deeper models (3 layers) tend to perform better, indicating that these rates require more model complexity to capture the underlying data patterns.

5. Conclusion

Comparative Analysis:

- The **Graph Neural Network (GNN)** model demonstrated superior performance in terms of cumulative returns, Sharpe Ratio, Sortino Ratio, and Maximum Drawdown when compared to traditional Hierarchical Risk Parity (HRP) models.
- The **sensitivity analysis** highlighted that the GNN's performance is highly sensitive to both the learning rate and the number of layers. A learning rate of **0.0005** paired with either 1 or 3 layers consistently produced high Sharpe Ratios, indicating robust risk-adjusted returns.
- The GNN model showed that deeper architectures (3 layers) perform better, particularly when using lower or higher learning rates (0.0001 or 0.0010), suggesting that model complexity is beneficial in capturing intricate market relationships.

Implications for Portfolio Management

1. **Enhanced Performance:**
    - The GNN model's ability to generate higher Sharpe and Sortino Ratios indicates a more efficient risk-adjusted return profile. Portfolio managers can leverage this model to optimize portfolios with better downside protection (as indicated by the Sortino Ratio) and minimized drawdowns, crucial in volatile markets.
2. **Hyperparameter Tuning:**
    - The sensitivity to learning rates and layer depth emphasizes the importance of rigorous hyperparameter tuning. Portfolio managers implementing GNNs or other machine learning models should consider automated hyperparameter optimization techniques to fine-tune these settings for better model performance.
3. **Risk Management:**
    - The lower Maximum Drawdown achieved by the GNN model implies that it can better manage tail risk and protect against significant losses, which is critical for maintaining long-term portfolio stability. This feature makes the GNN model particularly appealing for managing risk in uncertain or bearish market conditions.

Future Work

1. **Exploring Other Machine Learning Models:**
    - Future research could explore other advanced machine learning models such as **Long Short-Term Memory (LSTM)** networks, **Reinforcement Learning**, or **Transformer-based architectures** to compare their effectiveness in portfolio management against GNNs.
    - Investigating ensemble methods that combine multiple models (e.g., GNNs with traditional models like HRP) could further improve robustness and performance.
2. **Improving GNN Architecture:**
    - Incorporating more advanced GNN variants, such as **Graph Attention Networks (GAT)** with multi-head attention, or **GraphSAGE** for inductive learning, could enhance the model's ability to generalize and perform better on unseen data.
    - Adding features like **temporal dynamics** (time series data) within the GNN framework could help capture the evolving relationships between assets, further improving the model's predictive power.
3. **Incorporating Alternative Data:**
    - Future extensions could also explore the integration of alternative data sources, such as social media sentiment, macroeconomic indicators, or even news feeds, to provide additional context and improve the model's predictions.
    - Combining fundamental analysis with technical indicators within the GNN could lead to a more holistic approach to portfolio management, potentially uncovering new alpha sources.
4. **Real-Time Portfolio Optimization:**
    - Implementing the GNN model in a real-time setting, where it continuously updates and adjusts the portfolio based on live data, could provide practical insights into its usability and effectiveness in dynamic markets.
    - Research could also explore **transaction cost modeling** and **slippage effects** within the GNN framework, providing a more realistic assessment of its performance in real-world trading scenarios.