Portfolio Optimization

Nicholas Wong
Massachusetts Institute of Technology
nicwjh@mit.edu

Abstract

Portfolio optimization refers to the quantitative method that helps investors choose the best mix of assets to achieve their investment goals. In this project, we propose a machine learning-based approach to portfolio optimization. We ensemble 4 supervised learning methods - Principal Components Regression (PCR), Random Forests (RF), Weighted Moving Average (WMA), and Gated Recurrent Unit (GRU) Neural Networks to predict NASDAQ-100 stock prices before optimizing our constructed portfolio using the Markowitz Mean-Variance framework to determine asset weights. In our analysis, we aim to develop a portfolio that combines cutting-edge machine learning methods with modern portfolio theory to achieve optimal return-to-risk for investors seeking controlled risk with strong returns. Through a sparsified optimization process, we present an optimization strategy that delivers superior risk-adjusted returns as compared to Standard and Poor (S&P) 500 and NASDAQ-100 benchmark portfolios over the same time horizon.

1 Introduction

Effective portfolio optimization is a cornerstone of modern finance where we aim to allocate assets in a way that maximizes return while minimizing risk - a trade-off that is measured by the Sharpe ratio. Traditional portfolio optimization methods, like the Markowitz mean-variance model [4], have been widely adopted to maximize return for a given level of risk. The classical mean-variance framework provides a foundational approach to the problem of optimal asset allocation by quantifying the trade-off between expected returns and portfolio variance.

The evolving landscape of data-driven decision-making and advanced machine learning techniques has opened new opportunities for developing more robust portfolio optimization strategies. In this work, we propose the combination of modern portfolio theory with cutting-edge machine learning models for asset return forecasting to construct optimized risk-to-return portfolios that outperform conventional benchmarks. Specifically, this project leverages an ensemble approach to predict asset returns and employs sparse optimization techniques to deal with issues of overfitting, investor preference for sparse portfolios, and high dimensionality. Through the integration of predictive analytics with portfolio optimization techniques, this work aims to optimize risk-to-return in our proposed portfolio and formulate interpretable investment strategies by sparsifying asset weights.

We demonstrate the effectiveness of our optimization strategy by benchmarking the optimized portfolio against standard market indices over the same time period. Experimental results show significantly higher risk-adjusted returns in our optimized portfolio as compared to the S&P 500 and NASDAQ-100 benchmarks.

2 Related Work

Prior works have explored the use of predictive models to enhance portfolio returns. Ban et al. in [2] and Ta et al. in [10] emphasize the integration of machine learning algorithms for return prediction, demonstrating that data preprocessing and technical indicators can significantly improve the accuracy

of these forecasts. Ban et al. highlight the value of incorporating domain-specific data features into machine learning pipelines [2], while Ta et al. show that even basic regression-based methods, when coupled with carefully engineered financial features, can outperform traditional statistical models in prediction accuracy [10]. These predictive advancements align with the increasing reliance on neural network architectures, as shown by Gunjan et al., who uses deep learning to capture nonlinear dependencies in stock price data [3].

A common challenge in portfolio optimization outlined in previous works is high-dimensionality. Specifically, high-dimensionality in the estimation and filtering of correlation matrices for large-scale portfolios. This issue is critical, as noisy correlation estimates can lead to suboptimal allocations. Tola et al. in [14] addresses this by employing clustering techniques and random matrix theory (RMT) to improve the robustness of correlation matrices, enhancing portfolio stability. This work provides a theoretical basis for handling high-dimensional data and complements more recent data-driven efforts.

Our research builds upon these foundational contributions by combining advanced predictive models with portfolio optimization under constraints. Specifically, in forecasting, we leverage deep learning architectures such as GRU models to capture temporal dependencies in tandem with more traditional statistical approaches for additional robustness. This ensemble approach addresses a key limitation of single-method forecasts: the potential for overfitting to specific patterns in the data. For instance, traditional approaches may struggle in volatile markets where historical averages fail to capture emerging trends. By contrast, the integration of multiple methods provides a balanced view, mitigating the weaknesses of individual models. The use of L_1 regularization in the optimization step introduces sparsity in the asset weight vector, providing a way to handle high-dimensional data. Furthermore, by benchmarking our optimized portfolio against standard market indices, this work evaluates the practical impact of our proposed asset allocation strategy. We contribute to the growing literature by integrating ensemble forecasting and constrained optimization into a cohesive asset allocation framework, bridging the gap between advanced machine learning methods and modern portfolio theory.

3 Data

We use stocks from the NASDAQ-100 index for constructing the optimized portfolio in this work primarily for their growth and volatility characteristics. The NASDAQ-100 index consists of mainly technology and growth-oriented companies that exhibit higher growth and volatility as compared to a broader market index, making it better suited given the project's aim of capturing dynamic price movements. These diverse price movements and trend shifts are able to enhance the robustness of the implemented forecasting models in capturing complex patterns. A full list of NASDAQ-100 tickers is available in the appendix at 5.

3.1 Data Scraping and Preprocessing

Historical stock data for NASDAQ-100 companies was sourced from Yahoo Finance using the *yfinance* API with a daily frequency. Data was limited to the time period from November 1, 2023 to November 1, 2024 in consideration of computational requirements of deep learning methods, inverting and multiplying large matrices, and merging substantial dataframes in the optimization step. For each of the 100 companies in the NASDAQ index, historical stock data was scraped containing the following features:

- Adjusted Close Price: Closing price of the stock adjusted for corporate actions (stock splits, dividends, etc.)
- Close, High, Low, and Open Prices: Final, highest, lowest, and first trading prices of the stocks during the trading day respectively.
- Daily Trading Volume: Total number of shares traded during the trading day, indicating the level of market activity.

Data preprocessing included multiple steps to transform the raw financial data from *yfinance* into a form suitable for portfolio optimization. Metadata and irrelevant rows were removed, date columns were standardized into a datetime format, and numeric columns (*Adjusted Close Price*, *Volume*) were coerced into appropriate data types with invalid and missing entries dropped.

3.2 Feature Engineering

Additional features representative of return and volatility were engineered to reduce the overall bias of our predictive models. These engineered features include:

- 20-Day Returns: Percentage change in adjusted closing prices over a rolling 20-day period.
- 20-Day Volatility: Rolling standard deviation of daily return over 20 days, reflecting stock price fluctuation.
- Normalized 20-day return and volatility: Standardized versions of the above engineered features with zero mean and unit variance.

20-day returns gives an indication of recent stock performance and is useful for identifying stocks with strong momentum, which is crucial for portfolio selection. Similarly, rolling volatility provides insight into changing market dynamics and provides useful information for adapting portfolio strategies to current risk environments. The 20-day period, an often-used proxy for the number of trading days in a month, provides insight into medium-term trends that are more actionable than short-term noise.

Missing values in these engineered features were imputed using mean imputation to ensure consistency across the dataset [8]. Key features were also standardized using *StandardScaler* to have zero mean and unit variance [9].

3.3 Exploratory Data Analysis

Trading volume shown at 6 has a highly skewed distribution (12.49 million mean volume, 1.14 billion maximum volume), indicating trading activity domination by large-cap stocks. The high standard deviation in *Adj Close* indicates significant variability in stock prices. Prices range from 6.71 to 4676.25, indicating the presence of statistical outliers or a few exceptionally high-price stocks. The substantial variability and skew in adjusted close prices shown underscores the importance of standardization in the preprocessing step to prevent the dominance of high-priced stocks in subsequent modeling steps. The time series evolution of adjusted close prices for the top 5 stocks by *Adj Close* in the NASDAQ-100 index at 3 is highly suggestive of a bull market.

4 Methods

We deploy four forecasting methods in Python - Weighted Moving Average (WMA), Principal Components Regression (PCR), Random Forests (RF), and Gated Recurrent Unit (GRU) neural networks for predicting prices of stocks within the NASDAQ-100 index. We use these predicted stock prices to compute predicted returns. Predicted returns are then used in a sparsified mean-variance optimization framework for asset allocation to determine asset weights. Lastly, relevant financial metrics (return, volatility, Sharpe ratio) of the optimized portfolio are computed and benchmarked against market-capitalization weighted standard market index portfolios covering the same time horizon.

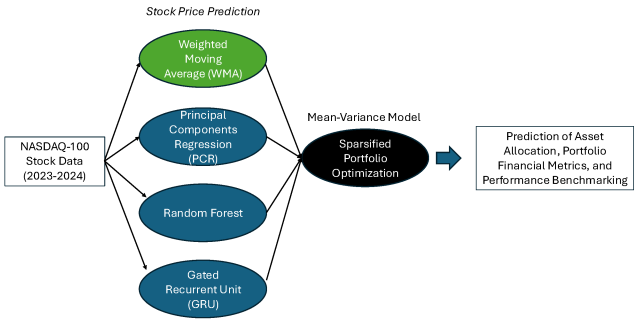

Figure 1: Project Architecture

For all forecasting methods, we employ a sliding window validation approach to evaluate model performance. For each method, we also compute predicted returns to be aggregated in the subsequent portfolio optimization process.

4.1 Weighted Moving Average

Weighted Moving Average (WMA) is a naive forecasting technique that assigns more weight to recent data and less weight to past data. The method's adaptability and simplicity make it particularly suited for financial time-series forecasting given the autocorrelation of financial data and need for robust yet interpretable modeling.

We implement a dynamic WMA model that predicts future stock prices using a rolling-window approach with a window size of 22-days (number of trading days in a month). Each forecast is generated as a weighted average of the most recent *n* days of adjusted closing prices, where weights are linearly assigned. Weights are defined as $w_t = t$ for the t-th day in the window to ensure the most recent data has the highest weight.

The WMA estimate for a given time t is calculated as:

$$\text{WMA}_t = \frac{\sum_{i=1}^{n} w_i \cdot P_{t-i}}{\sum_{i=1}^{n} w_i},$$

where P_{t-i} represents the adjusted closing price i days prior to t, and $w_i = i$ is the weight corresponding to the i-th day.

Future prices are forecasted through an iterative process:

- **Initialization**: Begin with the last n observed prices
- **Forecasting**: Calculate WMA and append to rolling window
- **Iteration**: Update the rolling window by removing the oldest price and including the new forecast

This process is repeated for the desired forecasting horizon in a stepwise forecasting fashion. Short-term forecasts are generated iteratively across the entire dataset where predictions rely on local trends within the 22-day historical window to forecast trends over the next 22-day horizon. This method allows us to capture recent trends and adjust dynamically as new data becomes available. Given the non-stationary nature of financial data, a sliding window approach is more appropriate as it allows the model to focus on a localized segment of the data rather than assuming a single global distribution for the entire dataset.

4.1.1 Sliding-Window Validation

To evaluate predictive performance of WMA and the rest of our forecasting methods, we employ a sliding-window validation approach. Given the temporal dependencies inherent in stock price time-series data, classical validation set and k-fold cross-validation approaches are inappropriate for this work. Sliding-window validation allows us to preserve the temporal structure of financial data, adapts to non-stationary market dynamics, and is able to evaluate the model's robustness across multiple future periods. It also aligns closely with real-world forecasting practices where predictions are constantly updated based on recent data.

Price data is divided into training and testing windows:

- **Training Window**: Used to compute the WMA forecast over the past n days, where we use a window size of $n = 22$ trading days
- **Testing Window**: Compared the forecasted prices to the actual observed prices for the subsequent h days, where we use a forecasting horizon of $h = 22$ trading days

For each window, forecasts are generated for the testing period using the WMA method and the accuracy of these forecasts is quantified using Mean Squared Error (MSE), defined as:

$$\text{MSE} = \frac{1}{h} \sum_{t=1}^{h} \left(\hat{P}_t - P_t \right)^2,$$

where $\hat{P}_t$ and P_t denote the forecasted and actual prices, respectively. We further compute a normalized MSE to account for variations in stock price magnitudes across different assets. We normalize the MSE by the mean adjusted closing price:

$$\text{Normalized MSE} = \frac{\text{MSE}}{\text{Mean Price}^2}.$$

The normalization allows for a more fair comparison of predictive accuracy across the NASDAQ-100 dataset of diverse magnitudes.

These normalized MSE values are computed for each ticker, and the individual normalized MSEs for each of the 100 tickers are aggregated to compute an overall normalized MSE for the method:

$$\text{Overall Normalized MSE} = \frac{\sum_{i=1}^{N} \text{Normalized MSE}_i}{\text{N}}.$$

where $N = 100$ is the number of tickers analyzed. Overall normalized MSE provides an overarching view of the model's predictive performance across the entire portfolio of assets.

4.1.2 Predicted Returns

In addition to price forecasts, we compute predicted simple returns from the sequence of forecasted prices for each of our forecasting methods:

$$R_t = \frac{\hat{P}_{t+1} - \hat{P}_t}{\hat{P}_t}.$$

These returns serve as estimated returns of individual assets that are used in the subsequent portfolio optimization process.

4.2 Principal Components Regression

Principal Components Regression (PCR) is a hybrid method that combines dimensionality reduction through Principal Component Analysis (PCA) with linear regression to model the relationships between features and target variables. We opt to use PCR for its effectiveness dealing with high-dimensional datasets. In the project, we deploy PCR with a sliding window approach to forecast the adjusted closing prices of NASDAQ-100 constituents.

The PCR pipeline consists of three main stages:

- **Feature Selection**: We extracted the following features over a 22-day window: 20-day returns, 20-day volatility, and normalized versions of these features. These features were then standardized to ensure scale invariance before performing PCA.
- **Dimensionality Reduction**: PCA was applied to reduce the dimensionality of the feature space. The number of principal components retained was determined dynamically to ensure that a sufficient proportion of variance was explained.
- **Regression Modeling**: Multiple linear regression was performed on the transformed feature space to predict adjusted closing prices.

For each sliding window of $n = 22$ days, PCR is trained on the most recent 22-day window and produces a forecast for the next 22 trading days (forecasting horizon, $h = 22$). This forecast is represented as an output vector $\in \mathbb{R}^{22}$, containing the predicted prices for each of the 22 days. The process is repeated across the entire dataset using a step size of 22 days. Model validation follows the process outlined in 4.1.1.

4.3 Random Forest

Random Forest Regression (RF) is a robust ensemble learning method that constructs multiple decision trees during training to perform predictions by averaging their outputs. In this work, we utilize RF to predict the adjusted closing prices for a sequence of 22 trading days (forecast horizon, $h = 22$) based on engineered financial features. The non-parametric nature of RF allow it to model complex, non-linear relationships that are prevalent in financial time-series data.

Utilizing our four engineered features, we instantiate the RF model with 20 estimators (trees) and constrain the maximum depth to 5 to prevent overfitting. These hyperparameters were selected using a heuristic approach and small sample empirical testing. Grid search methods for hyperparameter tuning were not adopted for the *n_estimators* and *max_depth* hyperparameters due to computational constraints.

Validation is done with a sliding window validation framework, using the same process outlined in 4.1.1, but with the training window condensed to 15 days instead of 22 due to computational constraints when implementing RF on our data.

4.4 Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a type of recurrent neural network that is particularly adept at capturing temporal dependencies in sequential data, making it highly suitable for time-series forecasting in finance. Our four engineered features encapsulating momentum and risk characteristics of the stocks are fed as inputs for the GRU model to forecast the adjusted closing price target variable. We implement GRU with a training window size of $n = 20$ days and a forecast horizon of $h = 22$ days. The GRU model is constructed using Tensorflow[13] and Keras[12], comprising of the following layers:

- **Input Layer**: Accepts sequences of shape (20, features), where 20 is the sequence length and features represent the dimensionality of input data.
- **GRU Layer**: A single GRU layer with 16 units and ReLu activation to capture temporal dependencies and learn sequential patterns.
- **Dense Layer**: A fully connected layer with a single neuron to output the predicted stock price for the next day.

The GRU's architecture effectively leverages its gating mechanism to retain and discard information, making it suitable for noisy financial data. The model was compiled using the Adam optimizer with a MSE loss function[1]. Early stopping was also employed to prevent overfitting[11]. Validation follows the same sliding window approach outlined in 4.1.1.

Given the high computational cost of the GRU deep learning architecture and our substantial data size, several considerations were made to accommodate the computational complexity. Model architecture was kept relatively simple and the model was trained for only a single epoch per sliding window validation cycle to accommodate computational constraints. The batch size, controlling how many training sequences are processed in parallel, was dynamically adjusted to be the smaller of 16 or the available training samples.

4.5 Optimization Process

The portfolio optimization step follows a three-step process:

1. Estimation of Covariance Matrix
2. Predicted Returns Integration
3. Mean-Variance Optimization with L_1-regularization

Key performance metrics are then computed for the optimized portfolio.

4.5.1 Covariance Matrix Estimation

The portfolio optimization process begins with an estimation of the covariance matrix which quantifies the co-movements and interdependencies between asset returns. For any assets i and j, the covariance is defined as:

$$\text{Cov}(i, j) = \mathbb{E}[(R_i - \mu_i)(R_j - \mu_j)],$$

where R_i and R_j denote daily returns of assets i and j, and μ_i and μ_j denote their respective means. We construct the covariance matrix Σ from all pairwise asset covariances and save it for use in the subsequent optimization steps.

4.5.2 Integration of Predicted Returns

To integrate predicted returns from our forecasting models, we utilize predictions from all four models (WMA, PCR, RF, GRU) and average the predicted return for each ticker:

$$E[R]_i = \frac{1}{M} \sum_{m=1}^{M} \hat{r}_i^{(m)}$$

where M = 4 is the number of forecasting models and $\hat{r}_i^{(m)}$ is the predicted return of asset i from model m.

This method of calculating expected returns equally weights the forecasts from each of method to obtain an average predicted return for each ticker. Aggregating results from each of our methods has several distinct advantages:

- **Reduction in Model-Specific Bias**: Aggregating predictions mitigate biases and limitations of any single forecasting model, producing a more balanced and robust estimate.
- **Robustness to Noise**: Averaging across multiple methods dilutes the impact of overfitting, reducing sensitivity to noise and improving generalizability.
- **Incorporation of Diverse Perspectives**: Short-term trends, sequential dependencies, and nonlinear relationships captured in each individual method allows the portfolio optimization process to integrate complementary signals.

These aggregated expected returns for each asset i are combined to form an aggregated expected returns vector $\mathbf{E}$ to be used in the portfolio optimization process:

$$\mathbf{E} = \begin{bmatrix} E[R]_1 \\ E[R]_2 \\ \vdots \\ E[R]_n \end{bmatrix} = \frac{1}{M} \sum_{m=1}^{M} \begin{bmatrix} \hat{r}_1^{(m)} \\ \hat{r}_2^{(m)} \\ \vdots \\ \hat{r}_n^{(m)} \end{bmatrix},$$

4.5.3 Sparsified Mean-Variance Optimization

We implement sparse portfolio optimization using L_1-regularization to induce sparsity in the weight vector. Formally, we define the optimization objective function and corresponding constraints as:

$$\min_{\mathbf{w}} \ \mathbf{w}^\top \Sigma \mathbf{w} + \lambda \|\mathbf{w}\|_1$$

Subject to:

1. **Budget Constraint**:
$$\sum_{i=1}^{n} w_i = 1,$$
where $w_i \geq 0$ ensures a long-only portfolio.
2. **Target Return Constraint**:
$$\mathbf{w}^\top \mathbf{E} \geq R_{\text{target}},$$
ensuring the portfolio achieves at least the specified expected return (We set R_{target} = 10%, a proxy for the average S&P 500 annual return[6])

We utilize a grid search approach to tune the λ regularization hyperparameter, controlling the sparsity level, for $\lambda \in$ [0.01, 0.05, 0.1, 0.5, 1.0]. Sparse optimization is iterated over these values of λ to identify the optimal portfolio by Sharpe ratio using the Sequential Least Squares Programming (SLSQP) method from *scipy.optimize*[5]. λ = 0.05 is chosen as a result of this process of hyperparameter tuning.

Post-optimization, asset allocation weights for the optimized portfolio are determined, and key portfolio metrics are computed 6.

5 Results

5.1 Stock Price Forecasting

We present the overall normalized MSE (4.1.1) for the four forecasting methods:

Table 1: Overall Mean Normalized MSE for Forecasting Methods

Forecasting Method	**Overall Mean Normalized MSE**
Weighted Moving Average (WMA)	0.0071
Principal Components Regression (PCR)	0.5140
Random Forest (RF)	0.0058
Gated Recurrent Unit (GRU)	1.0286

Overall, if favoring parsimony, simple models like WMA may seem naive compared to more flexible machine learning methods but are highly effective for short-term forecasting. The non-parametric nature of RF allows it to capture non-linear relationships effectively, making it the best performing method overall. For stock price forecasting, RF and WMA should receive greater emphasis to minimize prediction bias.

GRU's high error underscores the need for more robust training (additional epochs, hyperparameter tuning) to leverage its sequential modeling capabilities. As a result, GRU has significant potential for improvement with additional training and hyperparameter tuning, making it viable for longer-term and more complex forecasting tasks out of sample if computational resources allow for rigorous hyperparameter tuning. PCR provides benefits from dimensionality reduction, but does not capture the complex nature of financial markets. Its reliance on dimensionality reduction and linear assumptions is anticipated to limit its out-of-sample performance, especially in volatile market conditions. Ensembling these forecasting methods for prediction, however, is still highly beneficial as outlined in 4.5.2. The ensemble approach we utilize in this work mitigates individual model biases and leverages complementary strengths to enhance robustness to noise.

5.2 Portfolio Benchmarking

We present the asset allocation weights for each stock in our optimized portfolio, obtained from our sparse portfolio optimization process, in the appendix at 6.

To provide a robust benchmark for evaluating the performance of the optimized portfolio, we calculated the same annualized financial metrics (return, volatility, Sharpe ratio) for the market-capitalization weighted S&P 500 and NASDAQ-100 market indices over the same time horizon from November 1, 2023 to November 1, 2024. This benchmarking process allows us to assess the relative performance of our optimized portfolio against widely adopted market indices.

We present the key financial metrics for our optimized portfolio and benchmark portfolios:

Portfolio	**Return**	**Volatility**	**Sharpe Ratio**
Optimized Portfolio	12.19%	0.67%	11.56
S&P 500 Benchmark	34.63%	12.25%	2.14
NASDAQ-100 Benchmark	35.63%	17.54%	1.58

Table 2: Comparison of Optimized Portfolio Metrics with Benchmark Portfolios

Results show highly favorable performance relative to the benchmark indices. Specifically, the optimized portfolio achieves a Sharpe ratio of 11.56, which is significantly higher than the S&P 500's 2.14 and the NASDAQ-100's 1.58. This demonstrates a superior risk-adjusted return in the optimized portfolio and reflects the success of the optimization process in balancing returns and volatility.

The annualized volatility of the optimized portfolio is also significantly lower than that of the S&P 500 and the NASDAQ-100. The reduction in portfolio volatility underscores the efficacy of the mean-variance optimization framework in diversifying away risk through optimal asset allocation while still achieving returns that exceed a set target.

The substantial improvement in the Sharpe ratio over the specified benchmarks highlights the advantages of active portfolio management over passive index tracking. Results demonstrate the value of quantitative optimization techniques in achieving superior portfolio metrics as measured by risk-to-return, emphasizing risk-adjusted returns over absolute return performance.

6 Conclusion

We propose to give greater consideration to WMA for stock price forecasting if favoring model parsimony, and recommend RF for forecasting longer time horizons or to handle time-series data with high volatility. The above outlined methods significantly outperform the other tested forecasting methods and are likely to generalize well out-of-sample.

In this work, we also present an implementation of a sparsified portfolio optimization framework that demonstrates significantly higher risk-adjusted return metrics as compared to market index benchmarks. However, the lower absolute return suggests a conservative investment approach where the optimization process prioritized risk minimization, which may not align with the return expectations of all investors.

Computational limits were a significant obstacle in this work given the substantial data size and high computational cost of deep learning methods. Long Short-Term Memory (LSTM) neural networks were initially adopted for forecasting stock prices in this project due to their well-established reputation for financial time-series forecasting [7], but the implementation was removed in favor of a simplified GRU model to save on computational cost when the RAM requirements to run LSTMs on our data locally proved unfeasible. GRUs are a streamlined version of LSTMs, simplifying the sequential architecture and reducing the risk of overfitting. With additional computing resources, more complex deep learning architectures could be adopted with a more rigorous hyperparameter tuning process beyond the simplified GRU implementation and hyperparameter heuristic approach we adopt here.

Potential extensions include an exploration of a portfolio rebalancing strategy. Developing a dynamic portfolio rebalancing framework to adjust portfolio weights in response to changing market conditions would enhance the robustness and adaptability of the investment strategy. Robustness checks, such as out-of-sample validation and backtesting, would also be useful to check that the performance metrics obtained are not artifacts of the specific dataset used. Moreover, there are multiple extensions that could be made to extend the practical trading applicability of this work. Since we forecast adjusted close prices instead of actual close prices, trading applications of this work are limited as adjusted prices include the effects of corporate actions. Adopting an implementation that forecasts actual close prices or returns would lend more practical trading applicability. Lastly, this work does not account for transaction costs. The incorporation of these factors would similarly lend more practical applicability to the proposed investment strategy.

References

[1] *Adam PyTorch 2.5 documentation — pytorch.org*. https://pytorch.org/docs/stable/generated/torch.optim.Adam.html. [Accessed 30-11-2024].

[2] Gah-Yi Ban, Noureddine El Karoui, and Andrew E. B. Lim. "Machine Learning and Portfolio Optimization". In: *Management Science* 64.3 (Mar. 2018), pp. 1136–1154. ISSN: 1526-5501. DOI: 10.1287/mnsc.2016.2644. URL: http://dx.doi.org/10.1287/mnsc.2016.2644.

[3] Abhishek Gunjan and Siddhartha Bhattacharyya. "A brief review of portfolio optimization techniques". In: *Artificial Intelligence Review* 56.5 (Sept. 2022), pp. 3847–3886. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10273-7. URL: http://dx.doi.org/10.1007/s10462-022-10273-7.

[4] Harry Markowitz. "PORTFOLIO SELECTION*". In: *The Journal of Finance* 7.1 (Mar. 1952), pp. 77–91. ISSN: 1540-6261. DOI: 10.1111/j.1540-6261.1952.tb01525.x. URL: http://dx.doi.org/10.1111/j.1540-6261.1952.tb01525.x.

[5] *Optimization and root finding (scipy.optimize); SciPy v1.14.1 Manual — docs.scipy.org*. https://docs.scipy.org/doc/scipy/reference/optimize.html. [Accessed 30-11-2024].

[6] *S&P 500 Average Return and Historical Performance — investopedia.com*. https://www.investopedia.com/ask/answers/042415/what-average-annual-return-sp-500.asp. [Accessed 30-11-2024].

[7] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "A Comparison of ARIMA and LSTM in Forecasting Time Series". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018, pp. 1394–1401. DOI: 10.1109/ICMLA.2018.00227.

[8] *SimpleImputer — scikit-learn.org*. https://scikit-learn.org/1.5/modules/generated/sklearn.impute.SimpleImputer.html. [Accessed 29-11-2024].

[9] *StandardScaler — scikit-learn.org*. https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html. [Accessed 29-11-2024].

[10] Van-Dai Ta, Chuan-Ming Liu, and Direselign Addis. "Prediction and Portfolio Optimization in Quantitative Trading Using Machine Learning Techniques". In: *Proceedings of the Ninth International Symposium on Information and Communication Technology - SoICT 2018*. SoICT 2018. ACM Press, 2018, pp. 98–105. DOI: 10.1145/3287921.3287963. URL: http://dx.doi.org/10.1145/3287921.3287963.

[11] Keras Team. *Keras documentation: EarlyStopping — keras.io*. https://keras.io/api/callbacks/early_stopping/. [Accessed 30-11-2024].

[12] Keras Team. *Keras: Deep Learning for humans — keras.io*. https://keras.io/. [Accessed 30-11-2024].

[13] *TensorFlow — tensorflow.org*. https://www.tensorflow.org/. [Accessed 30-11-2024].

[14] Vincenzo Tola et al. "Cluster analysis for portfolio optimization". In: *Journal of Economic Dynamics and Control* 32.1 (Jan. 2008), pp. 235–258. ISSN: 0165-1889. DOI: 10.1016/j.jedc.2007.01.034. URL: http://dx.doi.org/10.1016/j.jedc.2007.01.034.

Appendix

Table 3: Statistical Summary of Selected Features

Statistic	Adj Close	Volume	Normalized 20-Day Returns	Normalized 20-Day Volatility
count	25452.0	25452.0	25452.0	25452.0
mean	267.29	12490069.92	-1.40e-18	-8.38e-19
std	426.44	45372941.51	1.00	1.00
min	6.71	79400.0	-3.85	-2.74
25%	76.68	1442850.0	-0.63	-0.67
50%	156.35	3109600.0	0.00	-0.17
75%	270.76	7833000.0	0.61	0.37
max	4676.25	1142269000.0	4.98	5.49

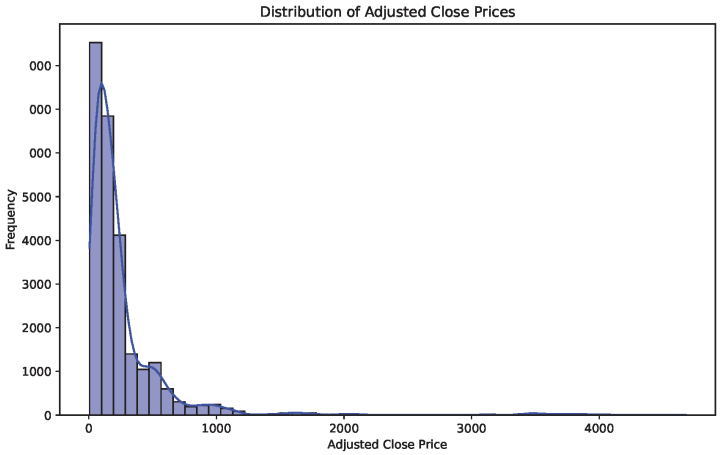

Figure 2: Distribution of Adj. Close Prices

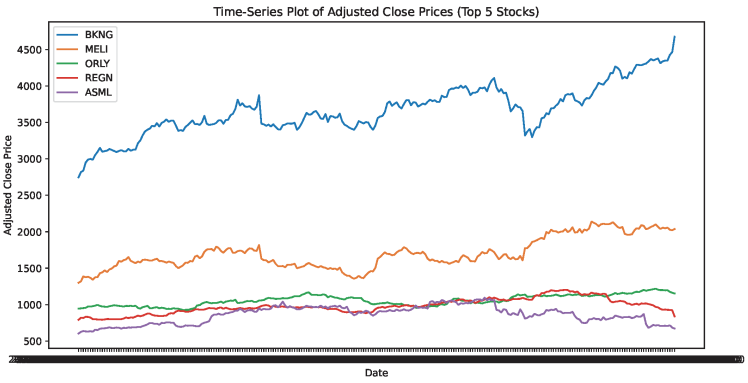

Figure 3: Time Series Plot of Adjusted Close Prices

Table 4: Portfolio Metrics

Metric	**Formula**	**Description**
Return (R_p)	$R_p = \mathbf{w}^\top \mathbf{E}$	Measures the expected return of the portfolio, where $\mathbf{w}$ is the vector of weights and $\mathbf{E}$ is the vector of expected returns.
Volatility (σ_p)	$\sigma_p = \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}$	Measures the risk of the portfolio, calculated as the standard deviation of portfolio returns using the covariance matrix Σ.
Sharpe Ratio	Sharpe Ratio $= \frac{R_p - r_F}{\sigma_p}$	A risk-adjusted return measure that compares the excess return of the portfolio relative to the risk-free rate r_F per unit of volatility.

Table 5: NASDAQ-100 Tickers

Ticker 1	**Ticker 2**	**Ticker 3**	**Ticker 4**	**Ticker 5**
AAPL	ADBE	ADI	ADP	ADSK
AEP	ALGN	AMAT	AMD	AMGN
AMZN	ANSS	ASML	AVGO	AZN
BIIB	BKNG	BKR	CDNS	CDW
CEG	CHTR	CMCSA	COST	CPRT
CRWD	CSCO	CSGP	CSX	CTSH
DDOG	DLTR	DASH	DXCM	EA
EXC	FAST	FANG	FTNT	GEHC
GILD	GFS	GOOGL	HON	IDXX
ILMN	INTC	INTU	ISRG	KLAC
KDP	KHC	LRCX	LULU	MAR
MCHP	MDB	MDLZ	MELI	META
MNST	MRNA	MRVL	MSFT	MU
NFLX	NVDA	NXPI	ODFL	ON
ORLY	PANW	PAYX	PDD	PEP
PYPL	QCOM	REGN	ROP	ROST
SBUX	SNPS	TEAM	TMUS	TTD
TSLA	TXN	TTWO	VRSK	VRTX
WBA	WBD	WDAY	XEL	ZS

Table 6: Sparse Portfolio Weights for Optimized Portfolio

Ticker	**Weight**	**Ticker**	**Weight**	**Ticker**	**Weight**
AAPL	0.0129	ABNB	0.0052	ADBE	0.0094
ADI	0.0034	ADP	0.0165	ADSK	0.0068
AEP	0.0194	AMAT	0.0000	AMD	0.0000
AMGN	0.0146	AMZN	0.0076	ANSS	0.0082
ARM	3.41E-19	ASML	0.0000	AVGO	2.88E-20
AZN	0.0183	BIIB	0.0136	BKNG	0.0103
BKR	0.0160	CCEP	0.0167	CDNS	0.0030
CDW	0.0109	CEG	0.0081	CHTR	0.0122
CMCSA	0.0155	COST	0.0145	CPRT	0.0114
CRWD	0.0034	CSCO	0.0144	CSGP	0.0106
CSX	0.0153	CTAS	0.0140	CTSH	0.0142
DASH	0.0063	DDOG	0.0040	DLTR	0.0133
DXCM	0.0133	EA	0.0151	EXC	0.0198
FANG	0.0172	FAST	0.0148	FTNT	0.0142
GEHC	0.0103	GFS	0.0000	GILD	0.0194
GOOG	0.0105	GOOGL	0.0105	HON	0.0151
IDXX	0.0087	ILMN	0.0048	INTC	0.0000
INTU	0.0079	ISRG	0.0100	KDP	0.0197
KHC	0.0202	KLAC	0.0000	LIN	0.0171
LRCX	3.12E-19	LULU	0.0083	MAR	0.0111
MCHP	0.0000	MDB	0.0005	MDLZ	0.0192
MELI	0.0104	META	0.0087	MNST	0.0179
MRNA	0.0033	MRVL	0.0000	MSFT	0.0124
MU	0.0008	NFLX	0.0122	NVDA	0.0000
NXPI	0.0004	ODFL	0.0190	ON	0.0000
ORLY	0.0190	PANW	0.0075	PAYX	0.0159
PCAR	0.0122	PDD	0.0107	PEP	0.0201
PYPL	0.0080	QCOM	0.0016	REGN	0.0154
ROP	0.0138	ROST	0.0137	SBUX	0.0122
SNPS	0.0025	TEAM	0.0037	TMUS	0.0195
TSLA	0.0000	TTD	0.0040	TTWO	0.0138
TXN	0.0065	VRSK	0.0188	VRTX	0.0161
WBA	0.0115	WBD	0.0065	WDAY	0.0110
XEL	0.0208	ZS	0.0016		