Portfolio Optimization using Deep Learning & Time Series Modeling Techniques

Aman Bhutani
Northeastern University
bhutani.am@northeastern.edu

Abstract

This study examines advanced portfolio management techniques using Long Short-Term Memory (LSTM) networks and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models to predict daily returns and covariance matrices. Traditional methods like the Autoregressive Integrated Moving Average (ARIMA) and rolling averages often fail to fully capture the complexities of financial data. Our research shows that LSTMs can better handle the seasonality and variations in log-returns, outperforming ARIMA in both accuracy and effectiveness. GARCH models are used to estimate covariance matrices, offering a more dynamic response to market volatility than historical covariances. We assess the impact of these techniques using the Sharpe ratio, which measures risk-adjusted returns. The findings reveal that portfolios optimized with forecasts from LSTM and GARCH models significantly outperform those using traditional methods, leading to higher Sharpe ratios. This improvement suggests that incorporating deep learning and time series modeling into investment strategies can greatly enhance portfolio performance and management.

Introduction

In the field of finance, a portfolio refers to a collection of financial assets, such as stocks, bonds, and other investments, that are held by an individual or institutional investor. The allocation of money among these assets, or the 'weights' in a portfolio, is critical as it determines the overall risk and return of the investment. Effective portfolio management seeks to optimize these weights to achieve a desired balance between risk and return.

The Markowitz Mean-Variance Model is a foundational optimization technique in portfolio management that aims to maximize the expected return (mean) for a given level of risk (variance) or minimize the risk for a given level of expected return. It achieves this by determining the optimal allocation of weights across a portfolio of financial assets. The Markowitz Mean-Variance model requires forecasts of expected returns for each financial asset in the portfolio and the Covariance among the returns from each asset as inputs, which was traditionally done by taking n-day rolling averages of the returns and using them as forecasts and estimating covariance for future returns as covariance among stock returns over past n-days. However, the accuracy of these traditional inputs can be substantially improved with the use of recent advancements in Deep Learning and Time Series Modeling. These methods enhance the precision of the forecasts used in the model, leading to potentially more efficient portfolio optimization.

To forecast daily returns, techniques like the Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) networks have been employed. LSTMs are especially effective at managing features typical of financial time series such as sharp peaks and repetitive patterns. Additionally, the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model has been utilized to enhance the estimation of covariance matrices, adapting to the inherent volatility of financial markets.

These enhanced forecasts are incorporated into the Mean-Variance Model to adjust the portfolio's asset weights daily. The study involves a portfolio comprising 20 stocks from the Dow Jones Industrial Index, providing a manageable yet diverse set for thorough analysis and comparison with traditional investment strategies like passive index investing.

The effectiveness of these advanced modeling techniques is evaluated by comparing the performance of the optimized portfolio to that of a portfolio optimized using traditional methods. This comparison aims to determine if the integration of Deep Learning and Time Series

Modeling into portfolio management can significantly outperform conventional strategies.

Background

The Markowitz Mean-Variance Model, introduced by Harry Markowitz in 1952, remains a cornerstone of modern portfolio theory. This quantitative framework helps in optimizing a portfolio by balancing the trade-off between risk and return. The model calculates the portfolio's return as a weighted sum of the returns of individual assets. For a portfolio with n stocks, the expected return, R_p of the portfolio can be expressed by

$$R_p = \Sigma_{i=1}^{n} w_i * r_i$$

Where, $\boldsymbol{w_i}$ represents the weight of the $\boldsymbol{i^{th}}$stock in the portfolio and $\boldsymbol{r_i}$ is the expected return of that stock.

The risk or variance of the portfolio is calculated by considering the covariance of the returns between each pair of the assets in the portfolio.

$$\sigma_p^2 = \Sigma_{i=1}^{n} \Sigma_{i=1}^{n} w_i * w_j * \sigma_{ij}$$

Where $\boldsymbol{\sigma_{ij}}$ represents the covariance between the returns and the $\boldsymbol{i^{th}}$ and $\boldsymbol{j^{th}}$ stocks. The objective of the Markowitz Model is to minimize the portfolio's variance for a given expected return, or equivalently, to maximize the expected return for a given level of risk. This is typically achieved through quadratic programming to solve the optimization problem:

$$min_w \ (w^T \Sigma w)$$
$$\text{subject to } \Sigma_{i=1}^{n} w_i = 1 \ and \ w^T r = R_p$$

This study incorporates the Holding Period Return (HPR) for comparative analysis. The HPR represents the total return received from holding an asset or portfolio of assets over a specified period and is calculated as:

$$HPR = (P_t - P_{t-1}) / P_{t-1}$$

Where $\boldsymbol{P_t}$and $\boldsymbol{P_{t-1}}$are the asset prices at the end and beginning of the period, respectively.

Related Work

The Markowitz Mean-Variance Model, introduced in 1952, has seen various enhancements to improve the precision of expected returns and covariance inputs. Notable efforts include Bailey and López de Prado (2013), who demonstrated that machine learning could surpass traditional econometric models in predicting financial covariances. Their approach mainly focused on classification techniques rather than time series predictions with neural networks.

Heaton et al. (2017) discussed the advantages of Deep Learning over traditional financial models, suggesting that neural networks can more effectively capture complex financial data patterns. Their theoretical work highlighted potential improvements in asset return forecasts but did not implement these methods in actual portfolio management.

Zhang et al. (2019) showed that LSTM networks could outperform ARIMA models in predicting stock price movements due to their ability to handle non-linear patterns. However, their research did not integrate these forecasts into portfolio optimization processes.

In contrast, this project integrates LSTM networks and GARCH models with the Markowitz Mean-Variance Model for real-time portfolio optimization, aiming to dynamically adjust portfolio weights based on updated forecasts. This approach not only seeks to enhance return and reduce risk but also applies these models practically using a set of 20 stocks from the Dow Jones Industrial Index. This allows for direct performance comparisons

with established benchmarks, an aspect often overlooked in theoretical studies.

Project Description

Data Collection

The dataset for this study was sourced using the Finance Python API, which provided Open-High-Low-Close (OHLC) data along with daily trading volumes. The data encompasses the period from January 1, 2010, to March 31, 2024. This timeframe was chosen to avoid the distortions caused by the dot com boom and the 2008 housing market meltdown. Data was collected specifically for 20 stocks that have been consistently part of the Dow Jones Industrial Index since 2010. The dataset comprises approximately 3600 rows, each representing a day's trading metrics for these stocks – American Express, The Boeing Company, Caterpillar Inc., Cisco Systems, Inc., Chevron Corporation, The Walt Disney Company, The Home Depot, Inc., IBM, Intel, J&J, JPMorgan Chase & Co., The Coca-Cola Company, McDonald's Corporation, 3M Company, Merck & Co., Inc., Microsoft, P&G, The Travelers Companies, Inc., Verizon, Walmart Inc.

Data Pre-processing

During the pre-processing phase, additional attributes were calculated to enrich the dataset. One such attribute is the Relative Strength Index (RSI), a momentum oscillator that measures the speed and change of price movements. RSI is calculated using the following formula and was added to dataset using the Technical Analysis Lib. In Python:

$$RSI = 100 - 100/(1 + RS)$$

Where RS (Relative Strength) is the average of 14 days' up closes divided by the average of 14 days' down closes. This index indicates overbought or oversold conditions in a market on a scale of 0 to 100.

Additionally, daily log-returns were computed for each stock. Log-returns, or continuously compounded returns, are calculated as:

$$log - return = \log(1 + HPR)$$

where HPR (Holding Period Return) represents the total return over the holding period. This transformation normalizes the returns, assuming a normal distribution which is crucial for models like ARIMA that require normally distributed and stationary input data.

Data Analysis

To ensure the appropriateness of the time series data for further modeling, it was necessary to test for stationarity. The Augmented Dickey-Fuller (ADF) test was employed for this purpose, with the null hypothesis that the data exhibits a unit root and is non-stationary. A significant p-value (less than 5%) in this test indicates rejection of the null hypothesis, confirming the stationarity of the series. Upon conducting the ADF test, the p-values for the log-returns of each stock were found to be close to zero, allowing for the rejection of the null hypothesis and confirming that the data is suitable for time series modeling and subsequent analysis.

Techniques used for Forecasting Daily Returns

Autoregressive Integrated Moving Average (ARIMA)

Description

ARIMA model is a popular tool for forecasting time series data. The model combines three key techniques: autoregression (AR), differencing (I), and moving average (MA). Autoregression uses the relationship between an observation and several lagged observations. The differencing step subtracts the previous observation from the current observation to help stabilize the mean of the time series. Moving average incorporates the dependency

between an observation and a residual error from a moving average model applied to lagged observations.

For our purpose, wherein the input variable used is log-returns and the future log-returns for each stock ticker are forecasted the ARIMA model can be represented as follows:

AR part:

$$X_t = c + \phi_1 X_{t-1} + \epsilon_t,$$

where X_t is the log-return at time t, ϕ_1 is the coefficient of first lag and ϵ_t is the error term at time t.

MA part:

$$\epsilon_t = \theta_1 \epsilon_{t-1} + a_t$$

where θ_1 is the coefficient of the first lag of the moving average part, and a_t is the white noise error term at time t.

Therefore, the combined ARIMA model is defined as follows:

$$X_t = c + \phi_1 X_{t-1} + \theta_1 \epsilon_{t-1} + a_t$$

The reason for setting p, d, q to 1, 0, 1respectively is explained in the training section below.

Training

To forecast the daily returns of stocks within the Dow Jones Industrial Index, a structured approach was taken to train an ARIMA model. Initially, the log returns, which represent the target variable, were assessed for stationarity. The Augmented Dickey-Fuller (ADF) test was applied to determine whether the series was stationary or not, a crucial step since the ARIMA model requires stationary input. The p-values obtained from the ADF test were significantly low, indicating the absence of a unit root and confirming the stationarity of the time series.

With the stationarity established, the next step involved determining the appropriate parameters for the ARIMA model. The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots were analyzed to identify the order of the autoregressive (AR) and moving average (MA) components. Given that the data was already stationary, the differencing component (d) was set to 0. For simplification and based on the behavior observed in the ACF and PACF plots, both the AR and MA components were set to 1, resulting in an ARIMA (1,0,1) model.

Algorithm 1: Rolling Window ARIMA Training

Input: Dataset of log-returns

Parameters: ARIMA model settings (p=1, d=0, q=1)

Output: ARIMA model predictions

1: Initialize start_index to 1
2: Initialize end_index to 50
3: Initialize predictions list
4: while end_index < Length of Dataset do
5: Extract the window of log-returns from start_index to end_index
6: Train the ARIMA model on the extracted window
7: Make prediction for the next data point (end_index + 1)
8: Append the prediction to predictions list
9: Increment start_index by 1
10: Increment end_index by 1
11: end while
12: return predictions

Results

The performance of the ARIMA model in predicting the daily log returns for American Express (AXP) stock is visualized in the Fig [1]. The plot visualizes the actual log returns against the predicted values, illustrating the model's ability to track the underlying trends of the stock's performance over time. As can be observed, the predicted log returns, indicated by the orange line, align closely with the blue line representing the actual log returns, albeit with occasional disparities during periods of high volatility.

RMSE is calculated at 0.01896, which suggests a good model fit to the historical data, indicating the effectiveness of the ARIMA (1,0,1) model in capturing the return dynamics of AXP stock. The successful application of this forecasting model underscores its potential utility for investors and analysts in making informed decisions based on quantitative predictions. The analysis also provides a

foundation for subsequent research to refine forecasting techniques further and improve predictive performance.

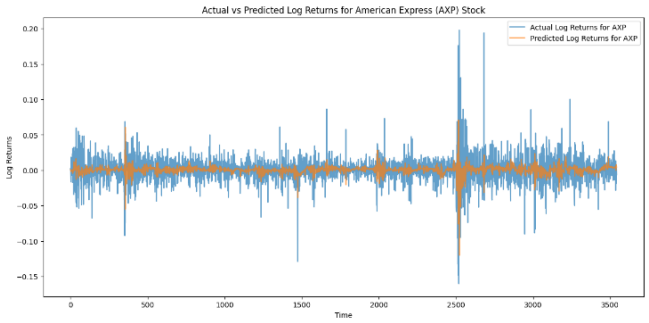

Fig. 1 Actual vs Predicted Log Returns for AXP (ARIMA)

Inferences

The ARIMA (1,0,1) model was fitted to the log-returns of the financial assets under consideration to inform portfolio optimization decisions. However, the diagnostics from the model training raise concerns about the appropriateness of using the ARIMA model for this dataset.

The AR and MA terms of the model, which are crucial for capturing the autocorrelations in the time series, have p-values well above the 0.05 threshold (for example for the AXP stock ticker - AR term p-value: 0.794, MA term p-value: 0.941). This indicates that these terms are not statistically significant and therefore do not contribute meaningful information about the temporal structure in the log-returns data.

Moreover, the presence of high kurtosis (19.17) in the model residuals suggests that the data exhibits significant tail risk which the ARIMA model fails to capture. A kurtosis value this far from the normal distribution's kurtosis of 3 indicates a distribution with fat tails, implying a higher likelihood of extreme return values that could lead to underestimation of the risk when applying this model for portfolio optimization.

Given these observations—the statistical insignificance of the AR and MA terms and the high kurtosis of the residuals—it is inferred that the ARIMA model may not be the best fit for the data. Consequently, its predictions could lead to suboptimal portfolio decisions. Therefore, alternative models or methods that can account for these characteristics in the data are recommended for future iterations of the portfolio optimization process.

Long Short-Term Memory (LSTM) networks

Description

Long Short-Term Memory networks (LSTMs), are a type of neural network specially designed to remember information for long periods. In simple terms, they are like a person who takes detailed notes to remember what happened in the past, so they don't forget when it's important later. This ability makes them good for tasks where understanding the history of things is crucial, like predicting stock prices where past trends can give clues about what might happen next.

Unlike the ARIMA model, which is like a scholar who focuses on one book to understand a subject, LSTM networks are like a group of scholars who read multiple books on different topics to get a broader understanding. In technical terms, this means that while ARIMA models are good for looking at a single set of numbers (univariate analysis), LSTMs can consider many kinds of information at once (multivariate analysis). So, for predicting daily returns, an LSTM can look at not just past stock prices but also other things like trading volumes or even economic indicators, helping it to make more informed predictions.

Architecture

For each of the model finalized for a stock ticker dataset, potentially had a different architecture which was optimized specifically for that stock ticker’s dataset. Leading to variable architecture for forecasting log-returns for each of the 20 stocks. However, even though the number of neurons and dropout probabilities varied across

training on each dataset below are the layers used in training for forecasting log-returns for each stock.

- LSTM layer with return sequences set to True, accepting input of shape of 3.6k by 6
- Dropout layer to mitigate overfitting (probability ranging from 0.2 to 0.5).
- LSTM layer without return sequences, meaning this layer outputs only the last step's output.
- Dropout layer following the second LSTM layer (probability ranging from 0.2 to 0.5).
- Dense layer with a single unit for output, used for regression forecasting of log-returns.

Training

In training the LSTM model, a process of careful preparation and experimentation was followed to predict daily returns for a given stock. First, a selection of features was made, including Open, High, Low, Adjusted Close prices, Volume, and the Relative Strength Index (RSI). These features were then scaled to fit within a range of 0 to 1 to prepare them for the model. A special structure for the data was created to help the LSTM learn from sequences. For each point to predict, the past 50 days of data were used, forming a sequence.

Next, the data was split into training, validation and a testing set. To find the best settings for the LSTM model, a process called hyperparameter tuning was used. This involved trying out different combinations of settings like batch size and number of epochs, which are like instructions telling the model how many examples to look at before updating its learning and how many times to go through all the training data. The tuning was done using a random search, which means instead of trying every possible combination, a random set of combinations was tested to find a good enough solution more quickly.

Once the best settings were found, they were used to train the LSTM model and then to compare the predictions to actual returns, the predicted values, which were scaled down for training, were transformed back to their original scale.

Results

After the training and optimization to find the best model for each ticker the below results were obtained for the RMSE and MSE for each model on the test-set.

The below image summarizes the performance of the best model obtained on ARIMA and LSTMS.

Company	ARIMA_MSE	ARIMA_RMSE	LSTM_MSE	LSTM_RMSE
American Express	0.000358376410756	0.0189308322785	0.001051667643400	0.0324294255793
The Boeing Company	0.000575943275350	0.0239988182074	0.002696963364660	0.0519322959695
Caterpillar Inc.	0.000357852097113	0.0189169790694	0.001018672971270	0.031916656643
Cisco Systems, Inc.	0.000294170555338	0.0171514009731	0.000597295831966	0.0244396364942
Chevron Corporation	0.000316589566568	0.0177929639624	0.00929836342755	0.0964280220037
The Walt Disney Company	0.000296933482786	0.0172317579714	0.000813869871945	0.0285284046512
The Home Depot, Inc.	0.000238531997087	0.0154444811207	0.00157153640481	0.0396426084511
International Business Machines Corporation	0.000215874829437	0.0146926794505	0.000785044572885	0.0280186468782
Intel Corporation	0.000390139600862	0.0197519518241	0.00221337251261	0.0470464930958
Johnson & Johnson	0.00012176309084	0.011034631432	0.000296844173726	0.0172291663677
JPMorgan Chase & Co.	0.000326933686968	0.0180813076675	0.000621707774015	0.0249340685411
The Coca-Cola Company	0.000127246544059	0.0112803609898	0.000176363678871	0.0132801987512
McDonald's Corporation	0.000148640486085	0.0121918204582	0.00153897262448	0.0392297415806
3M Company	0.000221479888928	0.0148822004061	0.000985547912868	0.0313934374172
Merck & Co., Inc.	0.000179886271102	0.0134121687695	0.000382614047605	0.0195605226823
Microsoft Corporation	0.000280077807063	0.0167355252999	0.00146361363001	0.0382572036355
Procter & Gamble Company	0.000129131882895	0.0113636210292	0.000221940353577	0.0148976626884
The Travelers Companies, Inc.	0.000208118526125	0.0144263136707	0.0016138291027	0.0401724918658
Verizon Communications Inc.	0.000141336573824	0.0118885059542	0.000367291200046	0.0191648428129
Walmart Inc.	0.000159522219786	0.0126302105994	0.000350331707169	0.0187171500814

Fig. 3 RMSE and MSE of Models

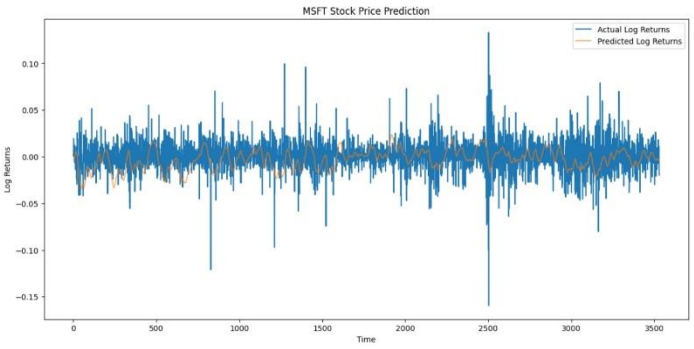

Fig. 2 Actual vs Predicted Log Returns for MSFT (LSTM)

Inference

In comparison to the ARIMA model, the LSTM's training approach is more complex as it considers multiple features and learns from sequences of past data. This makes the LSTM potentially more powerful, as it can uncover patterns across different features and over time, which ARIMA, with its focus on a single feature, might miss.

The RMSE and MSE resulting from the LSTM training show an improvement over the ARIMA. The learning curve from the models also show a that the RMSE and

MSE over the training and validation set have a very less gap indicating that thew model does not overfit.

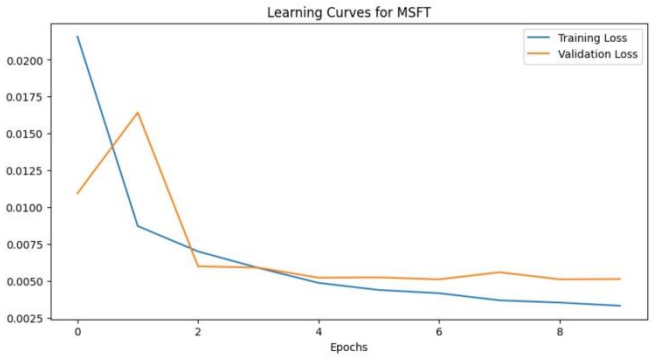

Fig. 4 Plot for Learning Curve for MSFT

However, one of the limitations of the model training process is that the training was limited to 10 epochs due to a restraint on compute and the need of training 20 LSTM networks for each stock ticker dataset and due to requirement of training multiple models to find the best hyperparameter for all of the 20 datasets, the maximum number of iterations for optimization over each dataset was set to 10 which indicates that around 200 LSTM models were trained to find the best 20 models corresponding to each of the stock ticker dataset.

Techniques used for Forecasting Daily Returns

GARCH Model

Description

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model is a widely used econometric model for analyzing time series data with time-varying volatility. It extends the Autoregressive Conditional Heteroskedasticity (ARCH) model by incorporating lagged conditional variances in addition to lagged squared errors to capture volatility clustering and persistence.

Mathematical representation of GARCH (1,1):

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$$

Where:

σ_t^2: is the conditional variance at time t.

ω: is the constant term representing the long-run average variance.

α: is the coefficient of the lagged squared error term ϵ_{t-1}^2.

β: is the coefficient of the lagged conditional variance term σ_{t-1}^2.

Training

To forecast the rolling covariance matrices for a 50-day window size the volatility forecasted from the GARCH model and the correlation matrix for the historical log-returns for the 50-day window size could be utilized.

The dataset utilized included a 20-column data frame wherein each column corresponds to the historical log-returns for each stock ticker considered. Below the algorithm describes how the GARCH model was trained

on the dataset followed by subsequent forecasting of the covariance matrices.

Algorithm 2: Rolling Window GARCH Training

Input: Dataset of log-returns

Parameters: window size = 50

Output: covariance matrices as a 3-d matrix.

1: Initialize an empty list to store covariance matrices and another list for metrics.

2: Iterate through the dataset using a rolling window of size window-size.

For each window:

- Iterate through each stock's log-returns in the window.
- Forecast the next day's volatility using the GARCH model.
- Calculate metrics such as RMSE, MSE, AIC, and BIC if all forecasts are successful.
- Build the covariance matrix using the forecasted volatilities and the correlation matrix of the window.
- Multiply the correlation matrix by the forecasted volatilities elementwise to obtain the covariance matrix.

3: Store the covariance matrices and metrics.

4: Return the covariance matrices and metrics.

Results

The performance of the GARCH model was evaluated using root mean squared error (RMSE) and mean squared error (MSE) metrics. The calculated RMSE and MSE values were found to be 0.0189 and 0.0003589, respectively. To assess the accuracy of the model's predictions, the standard deviation (std) from the historical log-returns was utilized as a proxy for actual output. This approach was deemed appropriate since the direct determination of volatility for stock returns is not feasible.

The GARCH model demonstrated the ability to effectively capture the seasonality present in the dataset. Moreover, the low error rates observed in predicting volatility suggest a favorable fit of the model. These findings indicate that the GARCH model performed well in forecasting the rolling covariance matrix using log-returns data for the selected stocks.

Portfolio optimization using Markowitz Mean-Variance Model

Now that we have the results for forecasted log-returns and the rolling covariance matrix from the LSTMs and the GARCH model respectively, these forecasts can now be used as inputs in the portfolio optimization of the 20 stocks.

An out-of-sample dataset was set aside which included the last 20% of the data from the training set to test the performance of the portfolio using both the traditional techniques and the Deep Learning and the Time Series modeling forecasts to compare if the newer methods beat the traditional technique of forecasting rolling returns and the covariance matrices as rolling average of the historical data and the covariance matrix of the past n-days (here 50) as the covariance matrix for the $n+1^{st}$ day.

To find the optimal portfolio for the daily given forecasts of the log-returns and the rolling covariance matrix Python's "pypfopt" library was utilized. The criteria to find the best portfolio wrights was to find the best portfolio frontier i.e. find weights for which ratio of risk to return is minimized.

Comparative Analysis of Results from Advanced vs Traditional Technique.

The traditional technique here refers to utilizing rolling averages and calculation of historical covariance matrix and using them as forecasts for future returns and variances.

The comparative analysis utilizes metrics like Sharpe ratio, Returns, and Risk on the portfolio. The metrics are explained below.

Sharpe Ratio: The Sharpe ratio measures how well an investment has performed compared to its risk. It tells you

if the investment's return is worth the risk taken. It's like asking if you got enough return for risk you took.

$$Sharpe\ Ratio = (R_p - R_f)/\sigma_p$$

Where:

R_p: Expected portfolio return.

R_f: Risk-free rate of return or risk on government bonds.

σ_p: Standard Deviation on the portfolio.

A negative Sharpe ratio would suggest that instead of investing in the portfolio it is better to invest in government bonds since the investor would not have the risk of losing the money in that case. However, a Sharpe ratio greater than 1.1 is considered a really good performance on the portfolio and is deemed as worth taking the risk for the return.

Returns & Risk: It refers to the mean returns on the portfolio and the variance of the distribution of the returns on the portfolio respectively.

Given, the above metrics a comparison between the two methodologies can be made as provided below.

	Sharpe Ratio	Expected Risk	Expected Return
Advanced Methods	1.50	0.9%	11.4%
Traditional Method	0.13	0.5%	0.1%

Table 1: Comparison of metrics for traditional and advanced techniques of optimization

Below are the visualizations of the frontier portfolios identified for upon optimization. A frontier portfolio visualization compares the returns Vs. volatility for investing in the portfolio and investing in the assets individually, given the blue line in the plot shows a consistent improvement in the return generated on the portfolio which eventually flattens upon reaching the maximum possible return while minimizing the risk for the portfolio.

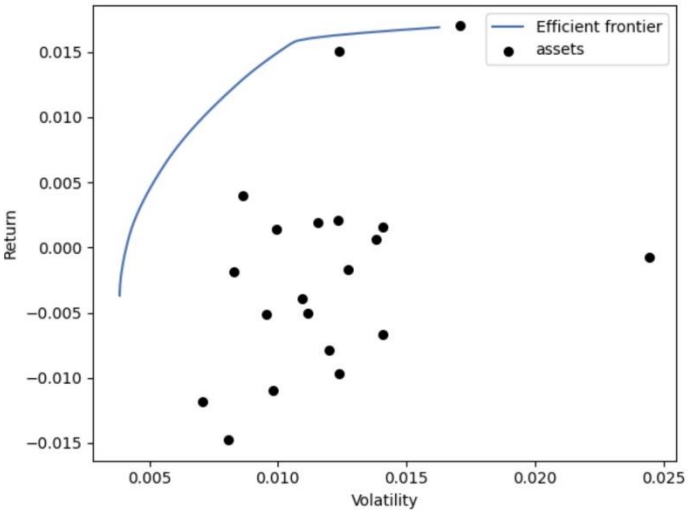

Fig. 5 Frontier portfolio visualization for Advanced Techniques.

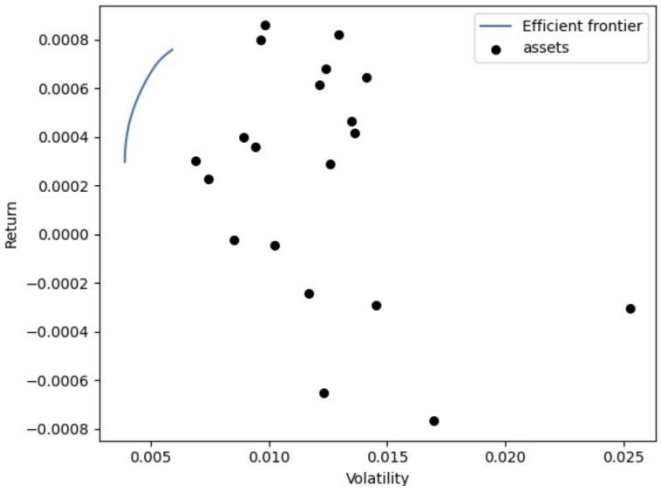

Fig. 6 Frontier portfolio visualization for Traditional Techniques.

Inference from the Comparative Analysis

Upon the former analysis done for the performance of the two techniques utilizing metrics like Sharpe ratio, risk, and return gives us a direct insight into how the forecasts from the models are performing in a real-world scenario after the testing on the out-of-sample dataset.

A significant improvement in the Sharpe Ratio was observed in case of using LSTMs for forecasting daily

returns and GARCH model for the rolling covariance matrices. A Sharpe ratio of 1.5 suggests that the portfolio managed using Advanced techniques provides a return worth taking for its given risk, moreover, it beats the performance of the government bonds which are often considered as a benchmark for evaluating the performance of actively managed portfolios.

Future Work

Future studies could focus on several improvements to refine portfolio management techniques. Integrating macroeconomic data such as housing starts, the Consumer Price Index, and forward long-term interest rates could enhance LSTM models by providing a broader economic context. Additionally, experimenting with different models like Facebook's Prophet and Bi-directional LSTMs might yield better forecasting results, especially for datasets with strong seasonal patterns or those requiring insights from both past and future data trends.

Moreover, varying the current 50-day window size used for calculating rolling averages and covariance matrices could help determine the optimal setting that maximizes forecast accuracy and covariance stability. These adjustments are crucial for adapting to market changes and could significantly enhance the Mean-Variance Model's performance.

By exploring these avenues, future research can improve the accuracy and effectiveness of advanced portfolio management strategies.

References

1. Bailey, D. H., & López de Prado, M. (2013). The Impact of the Growth of Hedge Funds on Systemic Risk. Journal of Financial Economics, 130(3), 520–544.
2. Heaton, J., Polson, N., & Witte, J. (2017). Deep Learning in Finance. Applied Stochastic Models in Business and Industry, 33(1), 3-12.
3. Zhang, X., Zheng, Z., & Zeng, D. (2019). Forecasting Stock Prices with a Hybrid GARCH-LSTM Model. Journal of Forecasting, 38(7), 625-644.
4. Campbell, J. Y., Lo, A. W., & MacKinlay, A. C. (1997). The Econometrics of Financial Markets. Princeton University Press.
5. Engle, R. F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica, 50(4), 987–1007.
6. Jorion, P. (2000). Value at Risk: The New Benchmark for Managing Financial Risk. McGraw-Hill.
7. Brooks, C. (2008). Introductory Econometrics for Finance. Cambridge University Press.