Portfolio Optimization Using LSTM Forecasted Returns

Hamad Alajeel, A16369878

[1]halajeel@ucsd.edu

Abstract— **The growth of both the financial markets and the computational capabilities of computers alongside the increase in accessibility of market data has led to great improvements in how institutions and individual investors are able to optimize portfolios. Using Historical data has been a major way in which institutions have optimized the weights of holdings in their portfolios, and theories which use statistical properties of historical data have aided in this process. One such theory is the Markowitz portfolio optimization theory which sets a number of assumptions regarding the definitions of risk and reward. However, portfolio optimization which only uses historical data is prone to much error, since the market is influenced by many factors, and historical data may fail to capture important trends in market prices of stocks. In this paper, we explore the possibility of using a Long-short-term-model (LSTM) to augment historical data with forecasted data for the final month and use the augmented data to optimize a portfolio of three stocks to maximize returns for that final month. A comparison is made between the returns of this portfolio and one which was only optimized using 3 months of historical data before the final month to compare their performances.**

I. Introduction

A. Markowitz Portfolio Optimization

We begin by discussing Markowitz portfolio optimization theory and how we can use it to optimize for portfolio returns. The Markowitz model is based on a set of important assumptions, and one of the most important assumptions in this theory is that risk is assessed based on the variability of returns from a given portfolio (1). This is why the Markowitz model is also called the mean-variance model because it uses the means and variances of the various portfolios it analyzes to determine the most optimal one (1). The objective of this framework is to maximize the expected reward of a portfolio given a unit of risk (2). This framework encourages diversification of a portfolio because it does not assess risk solely based on the individual risk and returns of holdings in a portfolio. It also looks at each holding's contribution to the overall risk of a given portfolio (2). In this model, portfolio return is the proportional weighted return of each asset given the weights of holdings in a portfolio, and the portfolio return volatility, or standard deviation of the risk of the portfolio, is composed of all pairs of correlation coefficients of the returns of the holdings in the portfolio (2). The equations are provided below:

$$\mathrm{E}(R_p) = \sum_i w_i \, \mathrm{E}(R_i)$$

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij}$$

Where the second equation of the portfolio variance can also be computed using (2):

$$w^T \Sigma w$$

where the *w* vector contains the weights of the holdings and Σ is the covariance matrix of the holdings' returns (2).

Knowing the portfolio's Risk and expected return, we also know the ratio of risk to return which is what the Markowitz model aims to maximize. It is also called the "sharpe ratio" and is a metric used widely by investors. This is the objective of our optimization problem and will be used to compute the most optimal weights for our portfolio of three stocks using both only historical data, and our hybrid historical/augmented data using forecasting by our LSTM model.

B. LSTM Modeling

LSTM is a type of RNN which is based on the idea of using data in the form of time series to capture and learn trends and then use that to predict values for future time steps (3). LSTM is designed to both retain and forget incoming pieces of input data to tweak the parameters it uses to predict the value of the next time step (3). LSTM's are composed of four components, the cell, input gate, output gate, and forget gate (4). Let's explain each component's function in order to make it clear why using LSTMs for stock forecasting is advantageous. The cell gate contains the current pieces of information retained from the previous state (4). The forget gate uses the input information to determine how much information to discard from the cell by mapping the input and cell information to a value ranging from 0 to 1 where 1 means remove all previous data, and 0 means retain all previous data (4). In a similar fashion the output gate determines which information is output using the information which was retained after the forget stage and the information from the previous state which was in the cell. Finally, the input gate determines how much of the new information should be stored in the cell (3). A simple diagram of an LSTM unit is shown below:

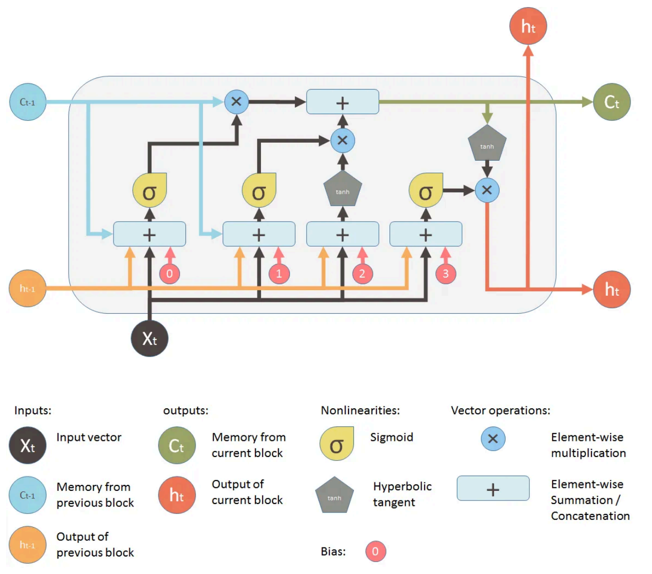


Figure 1: Diagram of LSTM Unit (5)

Therefore, for a correctly chosen RNN architecture which integrates LSTM units, we can use time series data to learn trends which allow the model to predict values of data for future time steps, and that is what we will implement in our experimentation set up.

II. Methods

For our experiment we have used 207 days of adjusted close prices for our three selected stocks, Microsoft (MSFT), Apple (AAPL), and Nvidia (NVDA), to calculate their daily returns and use that data for both forecasting and optimizing our portfolio distribution according to the Markowitz model. Daily returns are calculated using log returns which is the log of the adjusted close price of the current day divided by the adjusted closing price of the previous day for every single day of adjusted close prices for each of the stocks. Log returns are a useful tool for evaluating returns because they allow us to assess cumulative returns. Before, diving into how LSTM forecasting is done, basic information of the adjusted close prices of our three stocks are shown below:

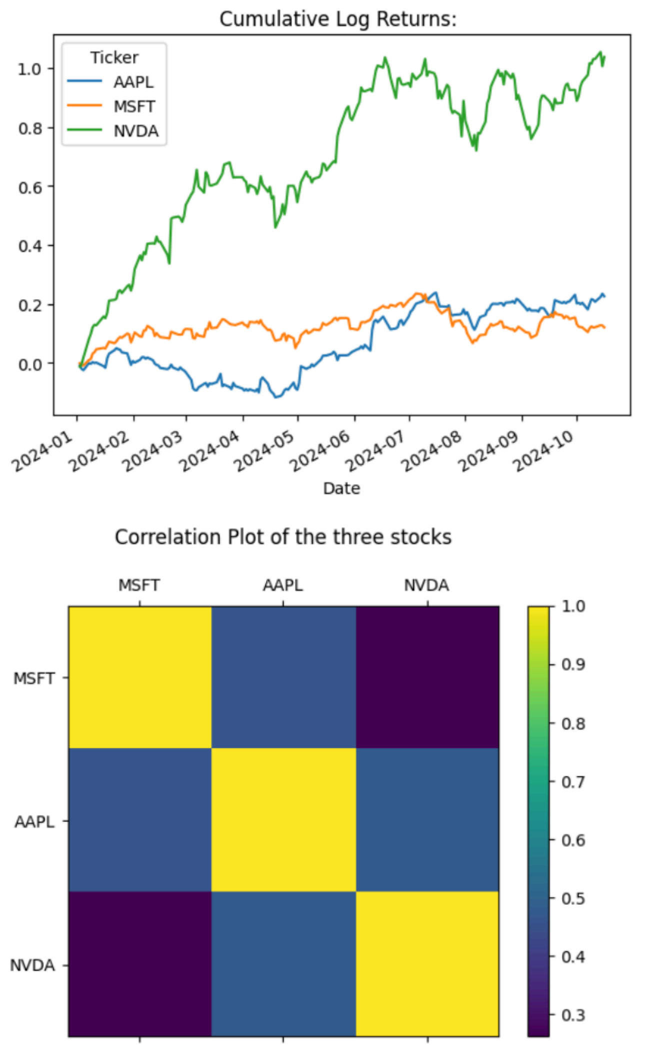
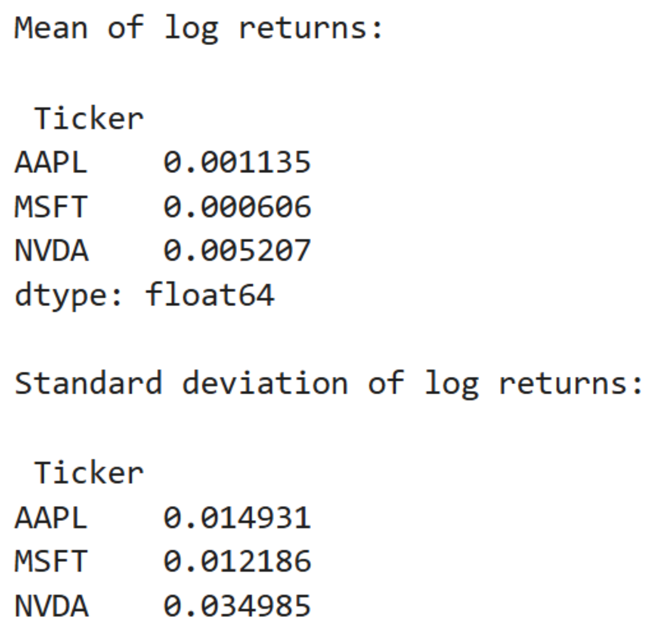


Figure 2: Cumulative returns, and Correlation Plots



```
Mean of log returns:

 Ticker
AAPL    0.001135
MSFT    0.000606
NVDA    0.005207
dtype: float64

Standard deviation of log returns:

 Ticker
AAPL    0.014931
MSFT    0.012186
NVDA    0.034985
```

Figure 3: Means and Standard Deviations of Log Returns

From these plots we observe that Nvidia's log returns are heading in an upwards trend, i.e. it is bullish, while Microsoft and Apple are wavering around low log returns. Additionally, Apple is more correlated with the other two companies than the other two companies are correlated to each other. The mean of Nvidia's log returns, reflects the same idea that it is outperforming the other two stocks by a large margin. With these observations in mind, one would invest in a portfolio with mostly Nvidia stocks and ignore the other two, so let us look at how a portfolio optimized using the Markowitz model would look like and compare. Using the data frame we have already set up, the sharpe ratio of the portfolio containing equal weights of these stocks is equal to 0.14. After passing the data frame along with a set of initial estimates of the weights of the portfolio to a Scipy minimization solver, we can maximize the sharpe ratio by minimizing its negation. After doing so, the optimal distribution of weights is 40% in Apple stock, 0% in Microsoft, and 60% in Nvidia. This gives us a sharpe ratio which is larger than that of equal weights which was 0.15 . This is a very intuitive solution, since we have already observed from the data that Nvidia is an outperformer, while Microsoft had the worst performance.

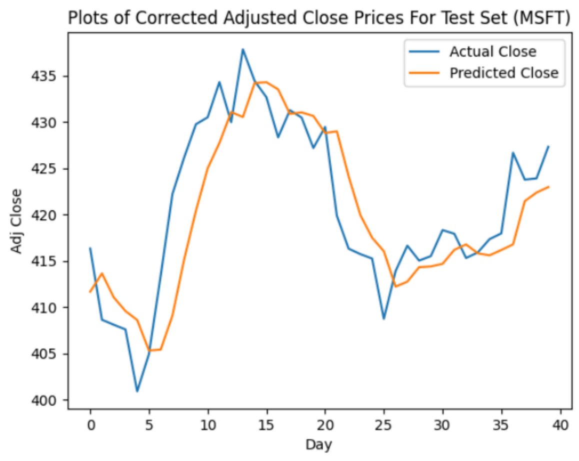
Now we discuss how we modeled our LSTM RNN. Our aim here is to use LSTM to forecast the adjusted close prices of our stocks for the final month of our data, compare that forecast to the actual closing prices, and then use that forecasted data for the final month along with historical data of the two preceding months to calculate the optimal weights for our portfolio by calculating log returns. Each of our 3 month of historical data preceding the final month and our 2-month historical data preceding the final month/ 1-month forecasted final month data will be used to compute optimal weights for our portfolio holdings, and their performances will be compared by evaluating their expected returns for the final month.

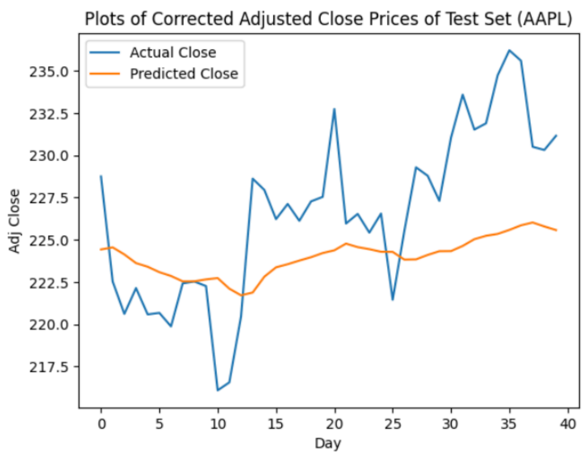
Three LSTM models were trained to forecast adjusted close prices for each of our stocks. The LSTM models' input units are 7-day time series data of the adjusted close prices of a stock on a given day. This means that our LSTMs learn to predict the adjusted close price on a given day based on the prices of the previous 7 days. Therefore, to generate time series data for each of the stocks, a matrix was created where each of the rows' first element represented the day we wished to predict the price for, and the rest of the entries were adjusted close prices of the preceding days. The first column in this matrix is our dependent variable, while the rest is data input to the model. In order to comply with our RNN model's required input size, we reshaped this matrix to be 3-dimensional where the final dimension size is 1. In the end, our train datasets have the shape of (number of days - 7, 7, 1), while our test dataset has the shape of (number of days - 7 ,1).

The LSTM RNN architecture is as follows: the input data is passed first through an LSTM module with input size 1, hidden size 4, and 1 stacked layer. This LSTM module is very simple, it is not stacked by multiple copies of itself. Then, its output is passed through a fully connected layer to generate predictions. The data was split into 80% training data and 20% test data. After careful fine-tuning, it was found that batches of size 20, and a learning rate of 0.01 resulted in good training. The loss function of the Mean squared error was used since this is a regression problem, and the Adam optimizer was used for gradient optimization.

III. Results

Below are plots for our predicted and actual adjusted close prices for each of the stocks for comparison:





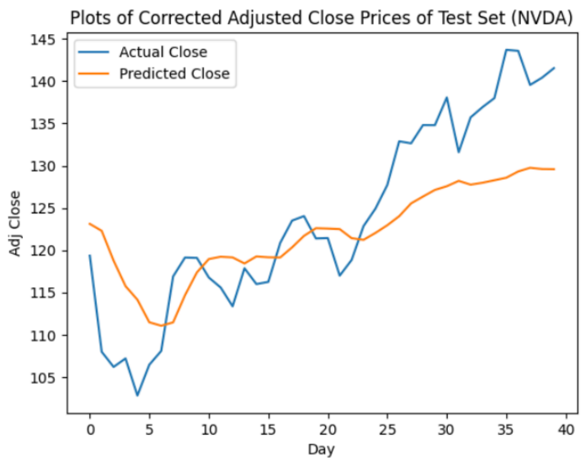
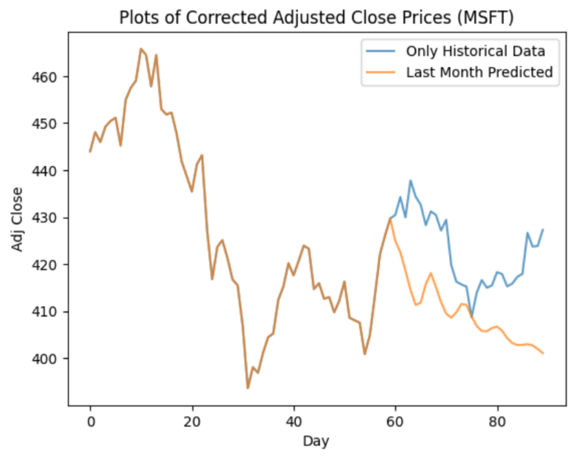
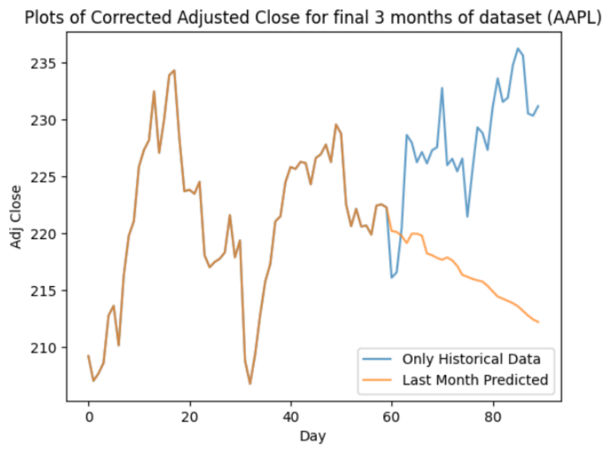


Figure 3: Test Set Predictions

The results shown above look promising, especially for the NVDA and MSFT adjusted close prices. However, to forecast the adjusted close prices for an entire month using our models, we cannot use data for the seven days preceding each of the days of that month, since that wouldn't be available in the first place. Therefore, in order to forecast prices for a whole month we can only do so recursively, by using adjusted close prices of the 7 days preceding the first day of that month, predict the price on the first day of that month, and repeat the process for the second day using the forecasted price on the first day and data for the days preceding it. At each step we append a vector of prices with a value predicted by our models based on the final 7 elements of this vector, i.e. the 7 days preceding the day we are predicting a price for.

After repeating this process with each of the models, plots of forecasted and actual close prices were provided for comparison:





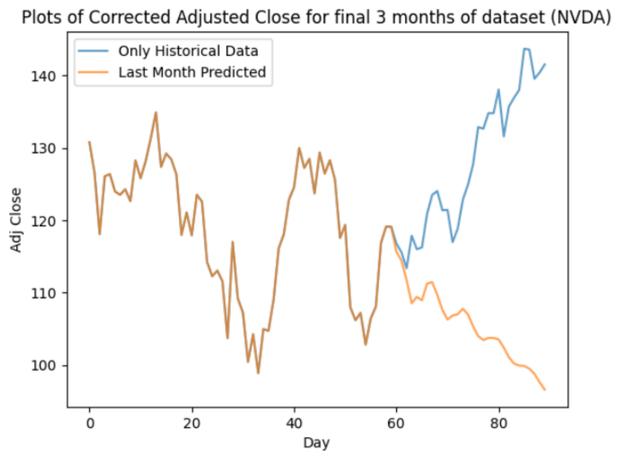


Figure 4: Final Month Forecasting Predictions

We see that in the case of forecasting prices for the final month recursively, our models failed to capture trends in our data, and bias in each case was very large.

After calculating log returns for the historical data of the 3 months preceding the final month in our data, and doing the same using our 2-month data preceding final month / forecasted final month augmented data, optimal weights were found for each of the datasets. The sharpe ratios for each of the portfolio weights were as follows: 0.19 for the only historical data, and 0.09 for our hybrid historical/forecast data. The expected log returns for the final month for each of the portfolio weights were 0.0023 for the only historical dataset and 0.0013 for the hybrid historical/forecasted dataset.

IV. Conclusions & Discussion

While in the case of predicting an individual day of prices given data for the 7 preceding days was quite successful, forecasting prices for a whole month led to completely inaccurate predictions which deviated very clearly from the direction of prices for each of our selected stocks. This led to weights which were optimized on inaccurate data leading to a low sharpe ratio and expected returns for the month we were interested in. Our model is quite rudimentary, both in architecture, and in the size and scope of data it trained on which is probably what led to huge bias when predicting values for a whole month. In order to solve this issue in future experiments, several points can be improved upon. Firstly, the model complexity should be increased to reduce the apparent bias in its predictions. Secondly, more data should be provided to the model, both in terms of the actual amount of data and the number of days our model looks back to predict the price of a stock for future time steps. It is also very likely that the objective of our model is too simple to predict prices on the scale of weeks or months. Perhaps, this model should be trained to predict prices for whole months given data of preceding months of many years, instead of simply a day to eliminate the need to predict prices recursively and to further reduce bias in our model.

References

[1] Wikipedia Contributors. “Markowitz Model.” Wikipedia, Wikimedia Foundation, 1 Sept. 2019, en.wikipedia.org/wiki/Markowitz_model.

[2] Wikipedia Contributors. “Modern Portfolio Theory.” *Wikipedia*, Wikimedia Foundation, 19 Mar. 2019, en.wikipedia.org/wiki/Modern_portfolio_theory.

[3] Aki Kakko. “Understanding Long Short-Term Memory (LSTM) for Investors.” Alphanome.AI, 12 Aug. 2023, www.alphanome.ai/post/long-short-term-memory-lstm-for-investors. Accessed 14 Dec. 2024.

[4] Wikipedia Contributors. “Long Short-Term Memory.” Wikipedia, Wikimedia Foundation, 22 Nov. 2018, en.wikipedia.org/wiki/Long_short-term_memory.

[5] Yan, Shi. “Understanding LSTM and Its Diagrams.” Medium, 15 Nov. 2017, blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714.