MF703 Final Project Report

Group 8

December 2023

Data preparation:

One of the important steps for stock prediction is data preparation. To achieve our goal of using machine learning technics to predict the trend of price dynamics for stocks, the historical data of stocks in the S&P 500 are considered. Some studies have been conducted by selecting 25 stocks from the SET 50 index as an experimental data set (Chaweewanchon, Apichat, and Rujira Chaysiri. 2022) for the machine learning prediction process before optimization. In this study, we selected 8 years of daily stock data. Specifically, we choose the data from 1 January 2010 to 30 December 2018 covering 2263 trading days to avoid the great volatility variation period during pandemic since it is easier to predict stable stocks than volatile stocks. This amount of data is sufficiently large for individual investors to build a portfolio (Zaimovic et al. 2021).

We choose 10 different industries(Technology, Healthcare, Financials, Real Estate, Energy, Materials, Consumer Discretionary, Industrials, Utilities, Consumer Staples, and Communications) to let our stocks represent the market as closely as possible. Then we pick 5 most representative stocks for each industry and put them in a stock pool, total of 50 stocks. After we set up our stock selection pool, we randomly chose 20 stocks as our

experimental data set to avoid the potential bias from selecting some highly correlated stocks. The names of these stocks are “AES Corporation” (AES), “UnitedHealth Group Inc”(UNH), “Apple Inc” (AAPL), “Procter & Gamble Co” (PG), “Prologis Inc”(PLD), “Digital Realty Trust Inc”(DLR), “Bank of America Corp”(BAC), “Thermo Fisher Scientific Inc”(TMO), “Johnson & Johnson”(JNJ), “Duke Energy Corp”(DUK), “Nike Inc”(NKE), “CVS Health Corp”(CVS), “Stanley Black & Decker Inc”(SWK), “Sherwin-Williams Co”(SHW), “Microsoft Corp”(MSFT), “DISH Network Corp”(DISH), “General Mills Inc”(GIS), “Mastercard Inc”(MA), “Conagra Brands Inc”(CAG), and “Best Buy Co Inc”(BBY). After we obtain the data for these stocks, we keep the open, close and volume data for our machine learning process. Also these data are been normalized to better fit the marching learning model.

Choosing model

Given the datasets that we have chosen, we need to make the decision on which model to choose. Upon research, we know that recurrent neural network (RNN) would be a suitable model, but it still has its own problem: the vanishing gradient problem.

When we train our neural network model, the model will learn based on the training dataset, and it will use back-propagation for weight update. The weight update function can be written as follows:

New weight = weight + learning rate * gradient.

Where the gradient is calculated based on the output, and find the derivative of the loss with respect to all of the inputs. The problem exists in that, the gradient is growing extremely small in very little time, so eventually gradient gets close to 0, and the new weight would basically be the old weight since if gradient = 0 then new weight = weight + learning rate * 0 = weight.

We don't want this to happen since otherwise the model won't learn. Hence, we found a solution, which is the Long Short-Term Memory model (LSTM).

LSTM, by the name of it, has longer memory and can store more information over a longer period of time. By the nature of LSTM, it is very powerful at learning Time-Series data, which is exactly what we have for our stock data.

However, LSTM has its pros and cons:

Pros:

-LSTM can handle long sequences of data since its cells can capture information from earlier time steps and remember it for a more extended period. Its memory cells also help us to learn better with past stock data.

-LSTM introduces the gating mechanics (forget gate, input gate and output gate) to control the flow of information, so the gradient would not become too small during back-propagation.

Cons:

-LSTM's architecture is more complex, so it would require more data to train. This is not a problem in our case, since we have enough history stock data to train our model.

-LSTM is not very resistant to noise in data. This is a problem that we haven't solved yet, and we will see in the latter part of our report.

So as we can see, LSTM is more suitable for our problem here, so we chose LSTM to predict the stocks' price.

-LSTM can easily overfit the training data.

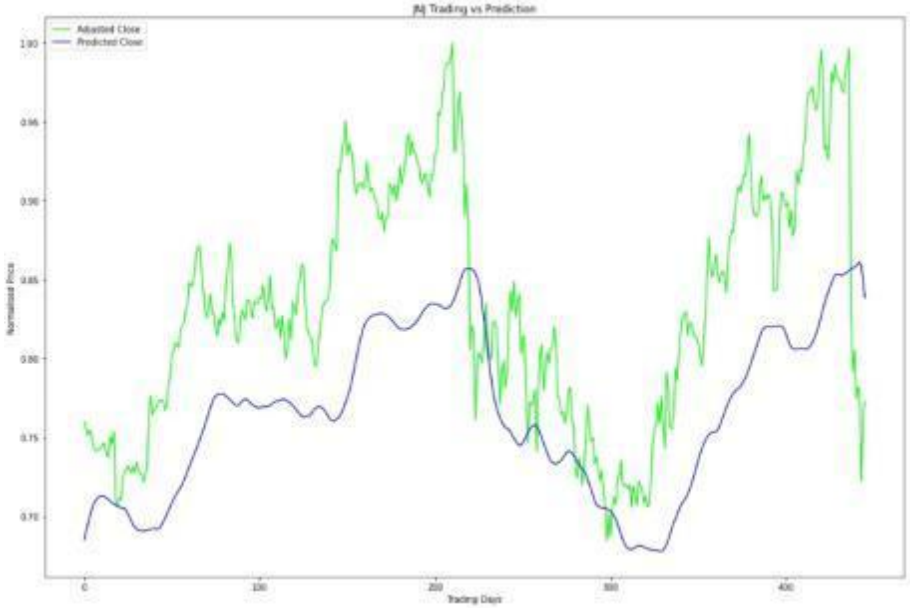
Implementing LSTM

Implementing LSTM is quite easy, as LSTM is already implemented for us in the Python Keras library. We can just use the library and import the model.

As for data preparation, we first clean and normalize the data, so that it has only opening price, closing price, and volume columns, and all of the data points are within the range of 0 and 1. We chose these 3 columns as our X-variables because, from our research and experience, these 3 factors are the most important in determining how the stock will go in the near future.

Next, we split the dataset into training and testing sets, by the 80-20 rule. When we train the model, we would use our 3 x-variables as our X-train set, and for the y-train, we decide to use the previous n days of data to predict the next closing price. So basically, if our training set has a length of 505, we would use train-ing_set[0:499] as our X-train, and training_set[5:504]["Closing price"] as our Y-train. We train our model this way, and it would be the same technique for our testing sets.

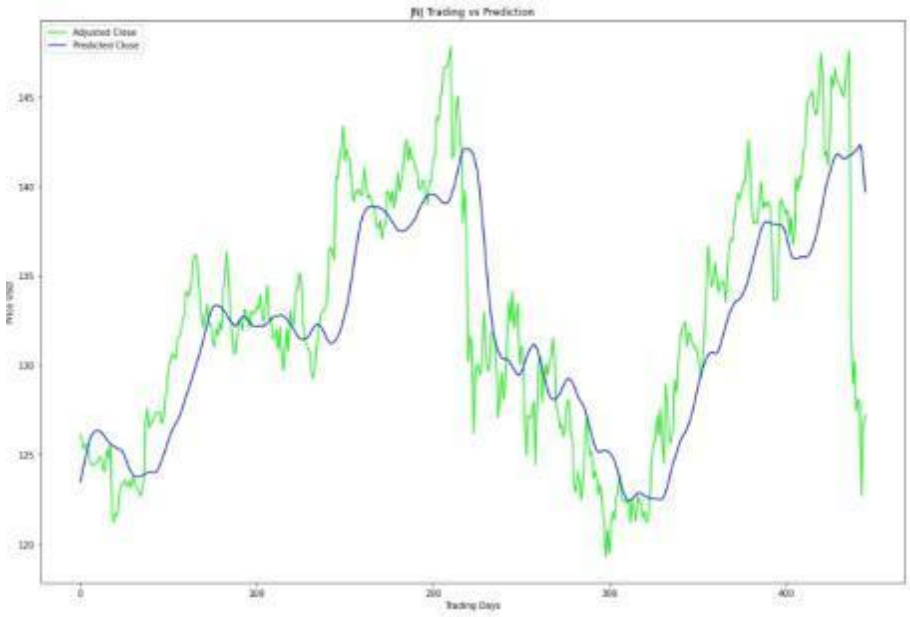
Finally, we simply throw the data into our trained LSTM model and get our predictions. This is our basic model, and we have our predictions visualized below:


AJ Trading vs Prediction
Adjusted Close
Predicted Close
Normalized Price
Trading Days

Improvements

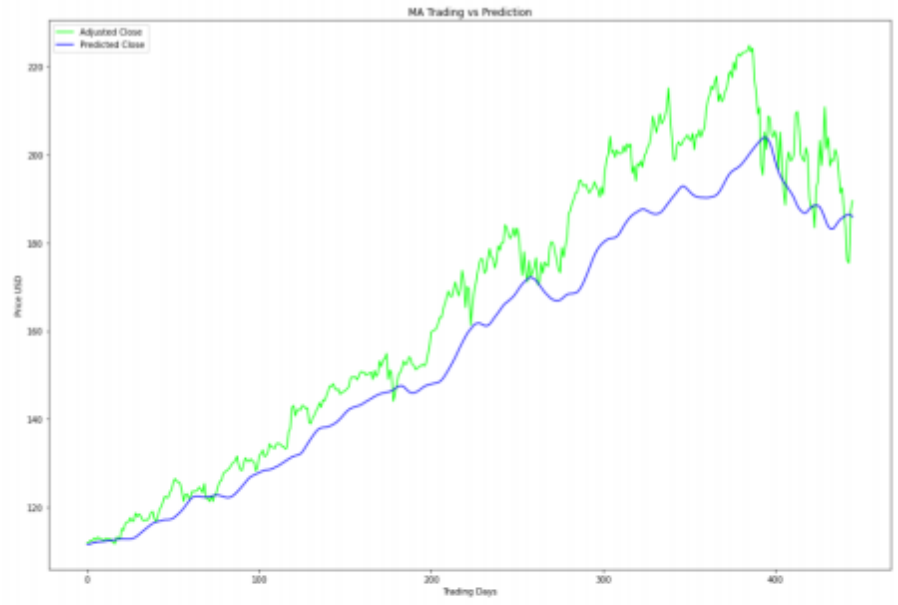
As we can see in the graph, our prediction follows the overall trend, but the actual values aren't so precise. We were not satisfied with this result, so we started considering how can we improve the model. The problem of over-fitting came to mind quickly since that's a very common problem in LSTM and can cause the predictions to be inaccurate. We want to test if overfitting is the problem here, so we used the dropout technique to solve the problem.

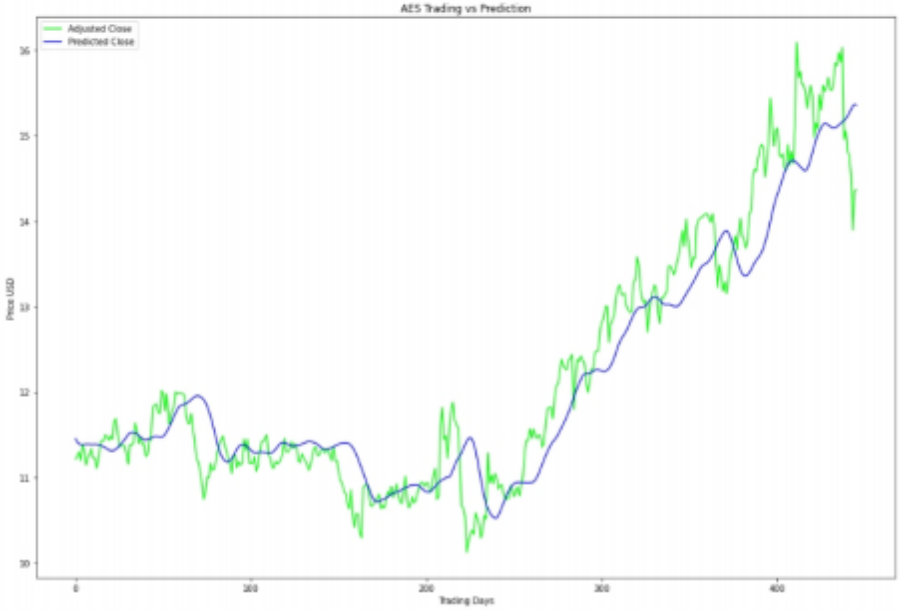
The dropout technique is a regularization method for LSTM, in which input and recurrent connections to LSTM are excluded from its activation function and gradient calculation while training the model. This is efficient in reducing over-fit, and thus improving performance. So we added a dropout layer between each layer of our original model, and let's see the result:



As we can see in the graph, our model's performance has improved drastically. To us, the new model can finally be used to predict an accurate price range. We are sure that there are still improvements to be made, but we were happy to see the result.

To make sure that the model works just fine for all stocks instead of just the one we chose, we ran the model over different stock data:


MA Trading vs Prediction
Adjusted Close
Predicted Close
Price USD
Trading Days


AES Trading vs Prediction
Adjusted Close
Predicted Close
Price USD
Trading Days

Yet, if we take a close look, near the very end of our first prediction on JNJ, there's a huge decrease in stock price happening in reality, and our model could not react to the drop in time. This is the problem of LSTM: if data noise like this happens, LSTM cannot predict such drastic change. We think this is not a problem about our model, but rather the nature of LSTM, or neural network models in general. We researched what happened during that time and realized that Johnson & Johnson had a huge baby powder problem near the

end of 2018. There were report articles on this event, so it came to us that, maybe if we build an NLP model to process articles before the stock market opens every day and combine that with our current LSTM model, it could solve this problem and give a more precise result. This is an improvement that we look to work with.

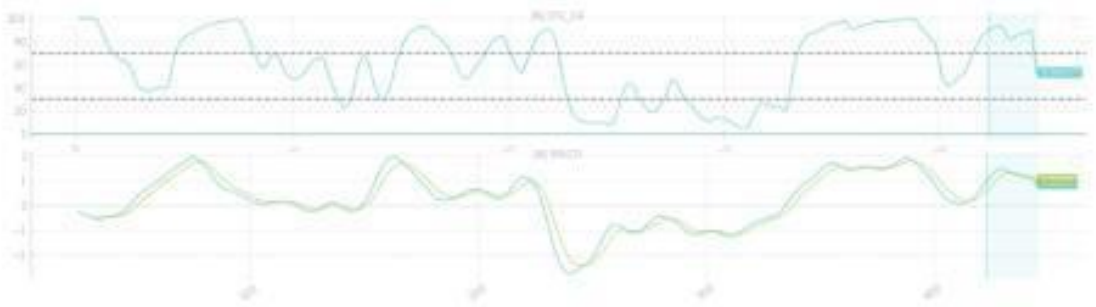
Stock Indexes

Merely predicting the stock price would not be enough. We want our program to give us suggestions on buying and selling points, so the users won't need to determine such points on their own.

To achieve such goals, we use stock indexes. Many traders would use the harmonic trading strategy, utilizing indexes such as MACD, RSI, BOLL, and techniques like Fibonacci and ABCD harmonic patterns. We decided to use MACD and RSI for our project since we are most familiar with these 2.

We know for a fact that, for MACD, when the MACD line crosses the MACD signal line from underneath, it would be a good buying point, and vice versa. We also know for RSI, if the index is underneath the 30 line and goes up, passing 30, then it's a good signal to buy, and if the index goes under 70 from above 70, then we know we should sell our stock.

We implemented our stock index strategies, and got the following outputs:



And the corresponding buying point and selling point.

Portfolio Optimization

Once all the stock prices are successfully predicted, high-quality stocks are selected to perform in the optimization process one by one by ranking them in descending order based on the expected annualized return, and ascending order based on the volatility.

As a result, we select the top (N) number of stocks with a higher potential return or a lower potential volatility according to the ranking order. Only the selected stocks are qualified for constructing the portfolio in the next stage. The MV model is used in this process to build the optimal portfolio with different proportions of asset allocation based on the qualified stocks. The optimization process is performed using the PyPortfolioOpt package in which the maximum sharpe is set as the objective function.

To comprehensively benchmark the machine learning selection model, we use random selection as a comparison.

This random model is different from the LSTM model in terms of stock selection. Specifically, in the first stage, the stock prediction is carried out randomly without

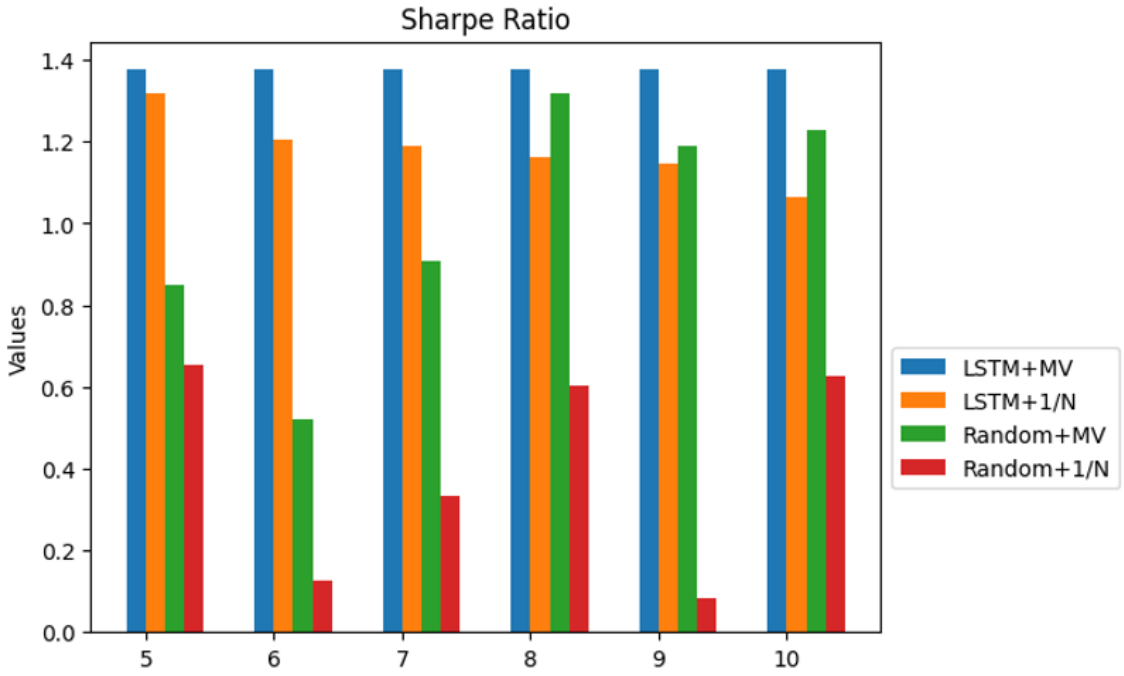
relying on any machine learning models. The stocks to be processed in the portfolio optimization using either the MV model or the 1/N model are selected randomly from all the samples in the second stage. The main objective of this strategy is to examine the necessity of stock selection using machine learning.

In this section, the performance of different optimal portfolios is measured and compared using three criteria, the Sharpe ratio, mean return, and risk of the portfolio. These metrics are widely used to evaluate and compare the performance of stock portfolios. The Sharpe ratio can be described as follows:

where Ep denotes the expected (average) return of the portfolio; σ is the standard deviation or risk of the portfolio; and Rf refers to risk-free assets. In this study, we use a risk-free asset rate of 0.04117, according to the 10-year US treasury rate.

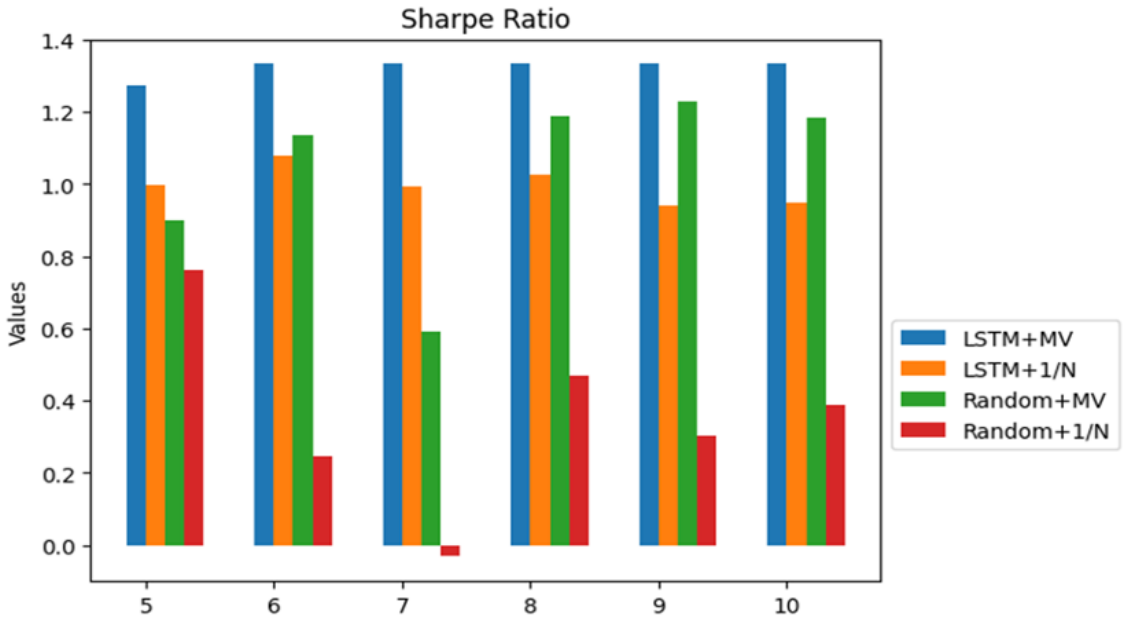
This study decides to construct portfolios corresponding to the number of stocks N = 5, 6, 7, 8, 9, and 10, and to comprehensively evaluate the performance of the proposed models, the annualized Sharpe ratio is employed as indicators.

For the result of the selection based on the descending order of the expected annualized return:



According to the figure, the LSTM+MV model outperforms other comparison models. However, there is a problem occurred in the results of the LSTM+MV model. When we rank based on return, the optimization tends to fix the weights on the earlier assets, resulting in the weights of the assets added later being mostly zero. This leads to little change in either the expected return or Sharpe ratio after adding more assets to the portfolio.

For the result of the selection based on increasing order of the annualized volatility:



This time, we scrutinize stocks with the smallest volatility predicted by the LSTM model against randomly chosen stocks. Notably, the LSTM+MV model consistently outperforms, yielding the highest Sharpe ratio, while the Random+1/N model remains at the bottom of the performance spectrum, as depicted in the graph.

An intriguing observation emerges when expanding the number of stocks in our portfolio. Notably, the green bar which is the Random+MV model consistently generates the second-best Sharpe ratio, closely mirroring the performance of the top contender. This trend is also evident in the figure depicting selection based on the descending order of expected annualized return. As the portfolio size (n) increases, the influence of the LSTM model diminishes, giving way to the dominance of the Mean-Variance (MV) model.

These findings hold practical significance, as investors often exhibit varying preferences when determining the number of stocks in their portfolios. In scenarios where a portfolio encompasses numerous stocks, the Random+MV model proves resilient, showcasing a performance akin to the best-performing model. This aligns with the reality that a larger portfolio may incur higher transaction fees but offers the advantage of diversification. Conversely, portfolios with a smaller number of stocks boast lower costs but carry an increased risk, especially during market downturns. This delicate balance represents a tradeoff rooted in investors' individual preferences.

In essence, the revealed pattern suggests that if investors opt for a larger, diversified portfolio or remain indifferent to the number of stocks, the Mean-Variance model suffices for portfolio construction. On the contrary, for those averse to high transition costs, the LSTM model assumes a more prominent role.

Citation:

Chaweewanchon A, Chaysiri R. Markowitz Mean-Variance Portfolio Optimization with Predictive Stock Selection Using Machine Learning. International Journal of Financial Studies. 2022; 10(3):64. https://doi.org/10.3390/ijfs10030064

Wei Chen, Haoyu Zhang, Mukesh Kumar Mehlawat, Lifen Jia Mean–variance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing Journal: 2020*