MACHINE LEARNING STOCK RETURN PREDICTION AND PREDICTION-BASED PORTFOLIO OPTIMIZATION

JUN CHANG
MASTER OF FINANCIAL ENGINEERING
CORNELL UNIVERSITY
jc2473@cornell.edu

SICHENG WANG
MASTER OF FINANCIAL ENGINEERING
CORNELL UNIVERSITY
sw863@cornell.edu

SHUHAN ZHANG
MASTER OF FINANCIAL ENGINEERING
CORNELL UNIVERSITY
sz444@cornell.edu

May 15, 2023

ABSTRACT

The purpose of this study is to investigate new methods for predicting time-series stock returns and to develop a new approach to Markowitz's portfolio optimization based on the predicted returns. To forecast stock returns, different time-series predictive models including ARIMA, Random Forest Regression, LSTM, and CNN-LSTM are implemented. Instead of the historical returns, the predicted returns obtained are fed into the Markowitz Portfolio Optimization to construct Mean-variance with forecasting (MVF) model. A significant number of simulations have been carried out on the 50 selected S&P 500 stocks. The findings suggest proposed MVF model based on ARIMA and CNN-LSTM predictions significantly outperforms the traditional Markowitz's portfolio in terms of both short-term return and Sharpe Ratio. Further research incorporating copulas into MVF is adviced.

***K*eywords** Time Series · LSTM · Convolutional Neural Network · Portfolio Model

1 INTRODUCTION

Portfolio optimization and asset allocation are critical tasks in the finance industry, with the primary goal of maximizing returns while minimizing risk. The Modern Portfolio Theory (MPT) proposed by Markowitz (1952) is a well-known method that has been used for decades to optimize portfolio performance. The benefit of using the MPT framework for Portfolio construction comes from diversification and the better trade-off between risk and return (Zhang et al., 2020). However, MPT heavily relies on historical asset returns and correlations, which have limited predictive power toward the future. In recent years, the application of machine learning algorithms in portfolio management has shown great potential in improving the accuracy of return predictions. By leveraging vast amounts of historical financial data and incorporating various technical indicators and other data sources, machine learning algorithms can generate more realistic predictions of future stock prices and returns compared to historical data.

This paper aims to explore the effectiveness of using machine learning algorithms to predict stock returns and subsequently incorporate these predictions into the MPT framework to optimize portfolio performance. We will discuss the benefits and challenges of using machine learning algorithms in portfolio management, as well as provide a comparative analysis of traditional MPT and MPT incorporating machine learning-based stock return predictions. Ultimately, this study seeks to demonstrate the potential of combining machine learning and MPT to improve portfolio performance and achieve superior investment outcomes.

2 LITERATURE REVIEW

The prediction of equity return is an intricate and complicated task. This is due to the financial time-series inherent noisy, non-stationary, and deterministically chaotic nature (Kumar and Thenmozhi, 2006). Researchers across the globe, from the past to the present, have been trying to find better ways to improve the forecasting accuracy of stock data. Financial time-series prediction can be broadly divided into two camps: the traditional statistical models and Neural-Network-based models (Gu et al., 2020). Traditional statistical time-series models include autoregressive integrated moving average (ARIMA) models, autoregressive conditional heteroskedasticity (ARCH) models, and generalized autoregressive conditional heteroskedasticity (GARCH) models. Machine Learning based time-series prediction models include Support Vector Machine (SVM), Random Forest, Recurrent Neural Networks (RNNs), and Deep Neural Networks (DNNs) (Rather, 2021). Machine Learning based models tend to outperform the traditional statistical time-series model due to the fact that Machine Learning based models can deal with data with more dimensions, and adapt to the non-stationarity and non-linear changes in the financial time-series data (Rather, 2021). Kumar and Thenmozhi (2006) utilized supervised Learning techniques such as SVM and Random Forest to predict the daily movement direction of S&P, CNX, and NIFTY index. Selvin et al. (2017) proposed a formal framework LSTM and CNN for stock price prediction. As a subunit of the RNNs, long-short-term-memory (LSTM) is frequently used for classifying, sorting, and making predictions from a single time-series dataset (Widiputra et al., 2021). One of the DNN

models that are frequently used to process multichannel input data is called Convolutional Neural Network (CNN) (Widiputra et al., 2021). Selvin et al. (2017) suggests that CNN architecture is capable of identifying the change in trends and CNN is the ideal choice for processing multivariate time-series data.

A portfolio optimization framework that can help achieve diversification of risk and excess return has long been one of the favorite areas for academic and industry research. Mean-variance analysis or MPT (Markowitz, 1952) is adopted by numerous institutional portfolios that derive portfolio weights by solving a constraint optimization problem (Zhang et al., 2020). The MPT constructs an efficient frontier, which represents the portfolio with the least risk for a given level of expected return (Ma et al., 2021). However, MPT suffers from some limitations such as restrictive hypotheses for potential risk and computational complexity for large portfolios (Ma et al., 2021). To address those issues, Alexander and Baptista (2002) proposed a model replacing mean-variance with mean Value-at-Risk (VaR), and Rockafellar and Uryasev (2000) developed a model using conditional Value-at-Risk (CVaR).

Classical portfolio optimization models based on MPT treat the mean historical return as the expected return. Such an approach tends to be able to capture the long-term stock return behavior via a low pass filtering (noise reduction) like mechanism (Ma et al., 2021). However, the backlash of treating mean historical return as expected return in MPT is the wildly inaccurate prediction of short-term stock/portfolio return due to short-term market volatility and investor sentiment (Ma et al., 2021). Thus, it is not sensible to use mean historical return as the short-term expected return. Stock return prediction should be combined with portfolio optimization methods to generate more reasonable short-term stock & portfolio returns (Ma et al., 2021). Many researchers utilized predicted return as expected return for portfolio optimizations (Ma et al., 2021). Freitas et al. (2009) adopted an autoregressive moving reference neural network to predict stock return in the Brazilian Market and used the predictive return as the expected return for the portfolio optimization. Ustun and Kasimbeyli (2012) even attempted to generalize the predictive optimization framework by using predictive returns to form objective functions in mean-variance analysis to improve the original Markowitz optimization model (Ma et al., 2021). Further, Yu et al. (2020) adopted the general framework and integrated ARIMA time-series forecast into several optimization models. Those integrated models demonstrated significant performance improvement from the original Markowitz optimization model (Ma et al., 2021). However, due to the limitations of the ARIMA model in dealing with Financial Time series (ex. normal distribution and linearity assumptions), Deep Learning models without restrictive assumptions demonstrate better performance when integrated into portfolio optimization (Ma et al., 2021). Rather (2021), Sen et al. (2021) applied an LSTM-based deep learning model for stock prediction and combined it with the portfolio optimization model which yielded high-precision results. Thus, it is interesting and crucial to look into the integration of deep learning prediction and classic portfolio optimization models (Ma et al., 2021).

3 METHODOLOGY

3.1 ARIMA

Auto-Regressive Integrated Moving Average (ARIMA) is a well-known time series model introduced by Box and Jenkins in 1970. ARIMA can be applied on any time series which are non-seasonal and has no random white noise (Khan and Alghulaiakh, 2020). According to Ariyo et al. (2014), ARIMA models demonstrated efficient capability in short-term forecasts outperforming complex structural models. The forecasting value of a variable in ARIMA consists of a linear combination of the past values and past forecasting errors (Khan and Alghulaiakh, 2020). The ARIMA(p, d, q) can be expressed as:

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$$

where Y_t is the actual current time value, ϵ_t is the random error at t, ω_i and θ_i are the coefficients, p stand for the order of the autoregressive part, d represent the degree of first differencing involved, and q is the order of the moving average part. The ARIMA model is suitable to forecast returns of well-diversified or non-volatile assets such as market ETFs or Large Cap blue chip stocks (Yu et al., 2020). Our study models the dynamics of the returns as an ARIMA(3, 1, 1) process. We use the previous 3 monthly returns to estimate the 4^{th} month's return. Since large-cap stocks tend to be less volatile than small-cap stocks and options, the returns of the 50 largest market-cap stocks within S&P500 are suitable for ARIMA forecasting.

3.2 RANDOM FOREST

Random Forest is a bagged ensemble learning method that is made up of multiple decision trees. In other words, the algorithm first grows multiple trees from bootstrap samples of the training data and randomly selected subsets of features. Then, by aggregating the results either by taking the average in case of regression or the majority vote for classification, it produces the final prediction. By clustering the results of multiple smaller trees, Random Forest is able to reduce variance while not increasing the bias very much. Moreover, as decision trees are able to capture nonlinear relationships between the independent and dependent variables, it is a powerful algorithm when it comes to predicting stock returns. Please refer to the diagram below for a visual understanding of the algorithm.

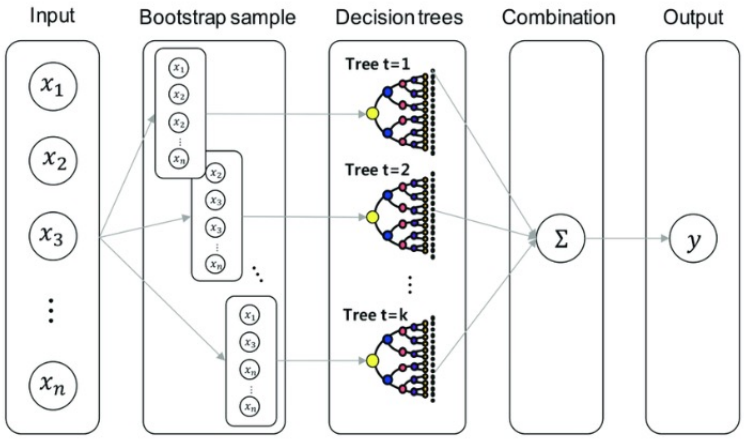

Figure 1: The internal structure of Random Forest (Kim et al., 2019)

3.3 LSTM

Long Short-Term Memory is a special branch of Recurrent Neural Network (RNN), which is time-recursive and is capable of prediction problems by learning order dependence with relatively long intervals and delays in time series (Zhang, 2022). It inherits the chain-like form of standard RNN, but instead of having a single $tanh$ layer like RNN, every repeating unit in LSTM consists of an input gate, forget gate, cell state, and output gate. This architecture allows LSTM to learn long-term dependencies as the cell state carries long memory of previous states' information and the gate mechanism optionally lets the information through.

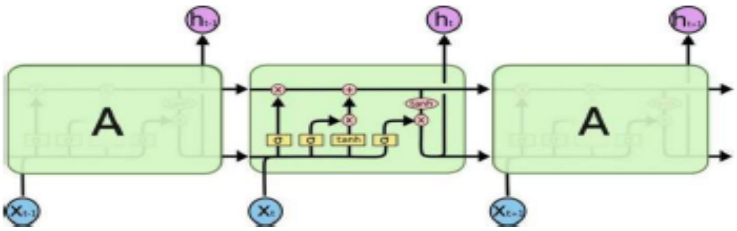

Figure 2: The internal structure of an LSTM (Zhang, 2022)

At the time t, the operation in each cell of LSTM can be expressed as the following (Bhandari et al., 2022):

$$i_t = \sigma(W_i x_t + W_{hi} h_{t-1} + b_i) \qquad (input\ gate)$$

$$f_t = \sigma(W_f x_t + W_{hf} h_{t-1} + b_f) \qquad (forget\ gate)$$

$$o_t = \sigma(W_o x_t + W_{ho} h_{t-1} + b_o) \qquad (output\ gate)$$

$$\tilde{c_t} = tanh(W_c x_t + W_{hc} h_{t-1} + b_c) \qquad (change\ gate)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c_t} \qquad (memory\ gate)$$

$$h_t = o_t \otimes tanh(c_t) \qquad (hidden\ gate)$$

where, σ and $tanh$ represent the $sigmoid$ and $hypoerbolictangent$ function respectfully; the operator $otimes$ is the element-wise product; $W \in R^{d \times k}$ and $W_h \in R^d$ are weight matrix, $b \in R^d$ are bias vectors.(Bhandari et al., 2022).

Finally, an LSTM model is constructed and fine-tuned for the prediction. The model uses monthly return data from 2013 to 2022 as the input. The model input shape of $(54, 1)$, which consists of a total of 54 features including the previous 3 monthly returns, the target return of the 4^{th} month, and 50 different dummy variables encoding 50 different stocks. The LSTM yields an output of shape $(54, 75)$ which suggests 75 features are extracted in each LSTM layer. After each LSTM layer, a drop-out layer is implemented. The drop-out layer randomly turns off 20% of the nodes in the LSTM layers to prevent overfitting (Bhandari et al., 2022). The structure of the LSTM can be seen in Figure 7 in Appendix.

3.4 CNN-LSTM

Convolutional Neural Network (CNN) is a DNN model that is widely applied in computer vision work and speech recognition (Sainath et al., 2013). Inspired by the biological processes, CNN's patterns of connectivity between neurons resemble the visual processing in cats (Shi et al., 2022). Previous studies such as Sainath et al. (2013) and Widiputra et al. (2021) suggest CNNs have superiority in analyzing time-based flowing data. As suggested by Widiputra et al. (2021), the main advantage of CNN lies in its local perception and weight-sharing features, which significantly shrink the number of parameters required and increase learning efficiency. CNN's structure is mainly made up of two parts: the convolutional layer and the pooling layer.

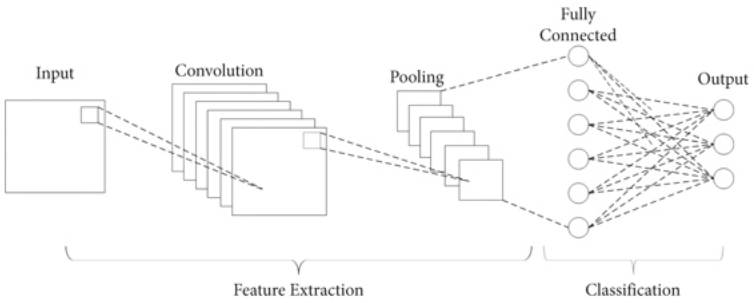

Figure 3: The internal structure of CNN (Widiputra et al., 2021)

(Widiputra et al., 2021) summarize the working mechanism of CNN that the convolution operation at the convolutional layer first extracts the most valuable features from the data followed by an increase in the feature dimensions. Then, to solve the problem of dimension increase, which can slow down the training process, an integration layer is added to reduce the number of features extracted before yielding the final result.

Deep learning techniques like CNN and LSTM give prediction outcomes with higher accuracy than other artificial neural networks according to the survey performed by Mahmoud and Mohammed (2021) on the usage of deep learning models for time-series forecasting. Moreover, Mahmoud and Mohammed (2021) had an interesting finding which

suggests combining multiple deep learning models greatly improved time-series prediction outcomes. CNN has poorer prediction accuracy compared to LSTM when applied to numerical time-series data. On the other hand, LSTM also suffers from its weak capability in extracting the most important features from data in contrast to CNN. Therefore, it is a sensible choice to construct a composite model that takes advantage of each combined model to overcome its weaknesses to increase the overall time-series prediction accuracy and generalizability (Widiputra et al., 2021).

For the purpose of this study, we architected an ensemble of CNN and LSTM, where the financial time-series data will be reshaped to match the input size that can be handled first by the CNN structure and then LSTM (Widiputra et al., 2021). The structure diagram and the schematic diagram of the CNN-LSTM model are presented in Figure 4 below. As we can see from the diagrams, the model consists of an input layer, a one-dimensional convolution layer, a max-pooling layer, a single LSTM layer, and a fully connected (Dense) layer which will yield the final forecasting value. The CNN-LSTM model training begins at the input layer as soon as the data enters. The training process continues at the convolution layer and pooling layer where the data is extracted of features and produces an output which will then serve as an input for the LSTM layer (Widiputra et al., 2021). When the data enters the LSTM layer, LSTM makes predictions mainly based on the observed time-series value. Subsequently, the output of the LSTM layer will enter the fully connected layer, which then produced the final prediction of the model (Widiputra et al., 2021). Like the LSTM model, the CNN-LSTM model has an input shape of $(54, 1)$, which consists of a total of 54 features including the previous 3 monthly returns, the target return of the 4^{th} month, and 50 different dummy variables encoding 50 different stocks. The one-dimensional convolution layer produces a three-dimensional output vector of (None, 4, 64) where the value 64 indicates the size of the convolution layer filters. The (None, 4, 64) vector then goes on to the LSTM layer where an output shape of (None, 64) is produced after training. The value 64 represents the number of hidden units in the LSTM layer (Widiputra et al., 2021). Figure 8 in the appendix depicts the schematic structure of the proposed CNN-LSTM model. Hyperparameters are tuned via trial and error. As a result, a batch size of 65 and epochs of 25 are used for training and validating the model. For the CNN layer and the LSTM layer, rectified line unit (ReLU) activation function is used. For the output layer, the sigmoid activation function is used instead. Mean squared error (MSE) is used to measure the training and validation accuracies (Sen et al., 2021).

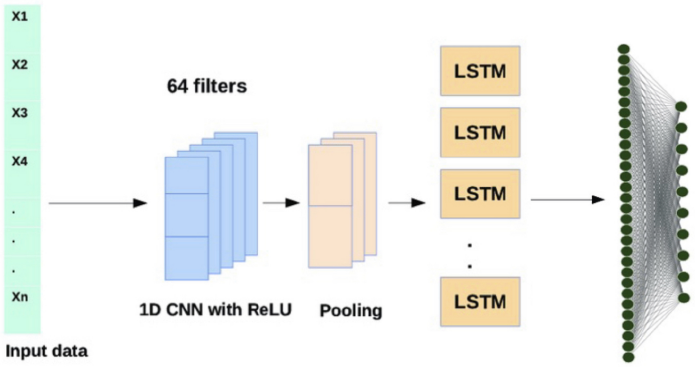

Figure 4: Architecture of the proposed CNN-LSTM model (Hamad et al., 2020)

3.5 Mean-variance with forecasting (MVF) model

Markowitz (1952) represents a major breakthrough in the field of portfolio optimization and diversification by introducing the MPT or mean-variance optimization (MV) model. The mean-variance presented a quadratic programming solution settling the trade-off between risk minimization and expected return maximization (Ma et al., 2021). The traditional way to estimate stock return is by taking the average of historical return, but this paper estimates the stock return by applying advanced machine learning algorithms, which we discussed above. Besides, as mentioned by (Zhang et al., 2020), the return forecasting approach can not always maximize the performance of a portfolio, we switch to optimize the shape ratio, which evaluates the maximizing return per unit of risk in our model.

$$\max \quad \frac{\sum_{i=1}^{n} w_i \hat{r}_i}{\sum_{i=1}^{n} \sum_{j=1}^{n} \sqrt{w_i w_j \sigma_{ij}}}$$

Subject to

$$\sum_{i=1}^{n} w_i = 1$$

$$0 \leq w_i \leq 1$$

$$i = 1, 2, ..., n$$

where w_i is the weighting of asset i in the portfolio, i is the number of assets in the portfolio, σ_{ij} is the element in covariance matrix, and $\hat{r}_i$ denotes the predicted return of asset i at time t (Ma et al., 2021).

4 MVF Portfolio Simulation

For the simulation data set, we have gathered the monthly returns of the top 50 market capitalization stocks in the United States from the beginning of 2013 to the end of 2022. Among the 10-year-long data, we have used 2/3 of it to train our base models and used the rest to extract the predicted returns of each stock and simulate the portfolio strategy by re-allocating our portfolio every month. To elaborate, we have substituted the expected returns (average of historical

returns), r_i, with the Machine Learning predicted returns, $\hat{r}_i$. However, for this study, like the original Markowitz, our model still used the covariance matrix generated from the past returns, which σ_{ij} is the covariance between stock i and j. Moreover, we have updated our Machine Learning models every three months and the statistical model (ARIMA) every month by feeding them with more recent data in a sliding window approach. As a benchmark, we have also simulated the portfolio strategy done under the traditional Markowitz optimization setting with the sliding window approach, in which the window size is the same as the training data.

The features used for the Machine Learning models are the returns of the past three months, the month of the return we are trying to predict, and 50 dummy variables that indicate which stock each row belongs to. Dummy variables are Boolean features that are recorded true if the data point belongs to the corresponding stock and false otherwise (one-hot encoding). By structuring our data in this manner, we have the advantage to build a singular model for the entire data set instead of training 50 different models for the 50 different stocks (Gu et al., 2020). Additionally, the ARIMA models were tuned separately for each individual stock, and the parameters for the ARIMA model were $p = 3, d = 1, q = 1$ in order to stay consistent with the features for the ML models.

5 RESULTS

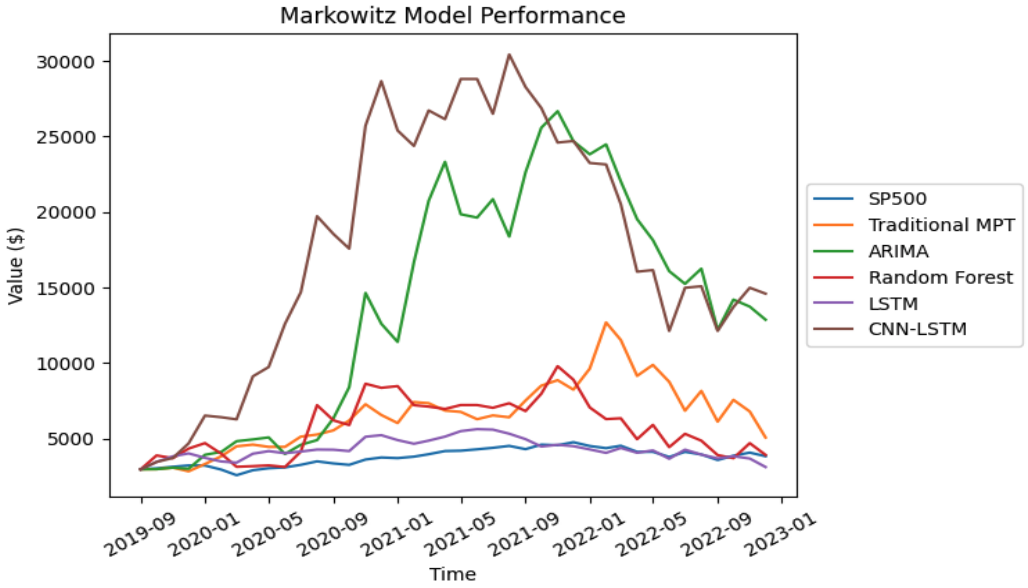

Figure 5: Portfolio Strategy Simulation

The figure above shows the results of the portfolio strategy explained previously. The starting values of the portfolios are equal to the value of the SP 500 on that day. Moreover, as a benchmark for comparison, the orange line indicates the value of the portfolio when done using the traditional Markowitz approach, and the blue line tracks the value of S&P 500 throughout the length of the simulation. It is important to note that only two methods (CNN-LSTM and ARIMA) performed better than the traditional method while Random Forest and LSTM under-performed. The superior

performance of the CNN-LSTM model can be attributed to the combined effect of CNN's feature extraction ability and LSTM's time-series predictive ability. Important to note that the performance of Neural Network models, such as LSTM and CNN-LSTM, may have varied output from time to time due to random weight assignment in the first iteration. Nevertheless, it is quite counter-intuitive how traditional linear regression model like ARIMA was able to outperform more complex model such as Random Forest and LSTM. This could be caused by the shorter updating period (ARIMA: 1 month; Other: 3 months). Nevertheless, the result still indicated ARIMA's robust short-term forecasting ability when facing non-volatile time-series data.

Method	RMSE	Average Returns	Variance	Sharpe Ratio
Traditional	0.0927	0.0234	0.0192	1.218
ARIMA	0.0952	0.0536	0.0362	1.482
Random Forest	0.0946	0.0246	0.0405	0.607
LSTM	0.0920	0.0046	0.0069	0.664
CNN-LSTM	0.0928	0.0547	0.0291	1.882

Table 1: Portfolio Strategy Statistics

From the table above, RMSE (Root Mean-Squared Errors) is used to evaluate the accuracy of the prediction of each model. The formula is

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y} - y)^2}{n}}$$

Where $\hat{y}$ is the predicted value of the returns and y is the true value. For the traditional method, we have substituted $\hat{y}$ with $\mathbb{E}[y]$, the historical average returns.

It is interesting to note that the accuracy of each model does not necessarily correspond to the performance of the strategy. For example, while LSTM had the most accurate model, it had the worst average return, but ARIMA, which had the worst accuracy, had the second-best average return. This could be due to the fact that the larger the residual or predictive error, $\epsilon_i = r_i - \hat{r}_i$, is actually corresponding to a higher abnormal return compare to benchmark (Yu et al., 2020). Overall, CNN-LSTM was still able to show the value of adopting deep learning into stock return prediction whereas the CNN-LSTM prediction-based portfolio was able to significantly outperform other prediction-based portfolios.

6 Conclusion and Further Work

This study attempted to extend existing research on machine learning-based time-series models for stock return prediction and portfolio construction with return prediction. With the objective to maximize risk-reward trade-off (sharpe ratio), one traditional regression model, one machine learning model, and two deep learning models are applied to improve the Mean-Variance Optimization model. This paper compares the predictive capabilities of ARIMA, Random

Forest, LSTM, and CNN-LSTM in stock return prediction. As shown in Figure 5 and Table 1, the portfolio based on CNN-LSTM prediction significantly outperformed other models and the benchmark in terms of both short-term return and Sharpe Ratio. However, the CNN-LSTM does suffer from an unstable mapping due to random initial weight assignment. Additionally, it is quite a surprise that the traditional regression model ARIMA was able to significantly outperform the more complex Random Forest and LSTM models, suggesting traditional model can also yield robust estimation when used in the proper contexts.

More work can be done to explore this study further. For future studies, we suggest incorporating factor models, adding average prediction residual maximization into the objective function, and integrating copulas for risk analysis. As we did not have many features in our Machine Learning model, additional predictors such as various factor models can be added to increase the accuracy of the prediction. In terms of improving objective functions, abnormal return maximization based on average prediction residual, $max \sum_{i=1}^{n} x_i \bar{\epsilon}_i$, tends to be effective in measuring abnormal returns and it can be added to the objective functions to help the machine learning model to capture additional abnormal return relative to the benchmark. Lastly, as the Markowitz model contain two parameters, return and variance, different multivariate distribution models such as copulas can be utilized to capture the correlation between different stocks to generate a more realistic forward-looking risk estimation compare to the covariance matrix. For instance, the performances of different stocks tend to resemble tail dependencies and to capture such characteristics, a Gaussian copula or t-copula could be a more suitable method. As a quick exercise, we have simulated a Gaussian copula-based MPT, where the covariance matrix is replaced by the fitted Gaussian copula correlation matrix. Please refer to the figure below.

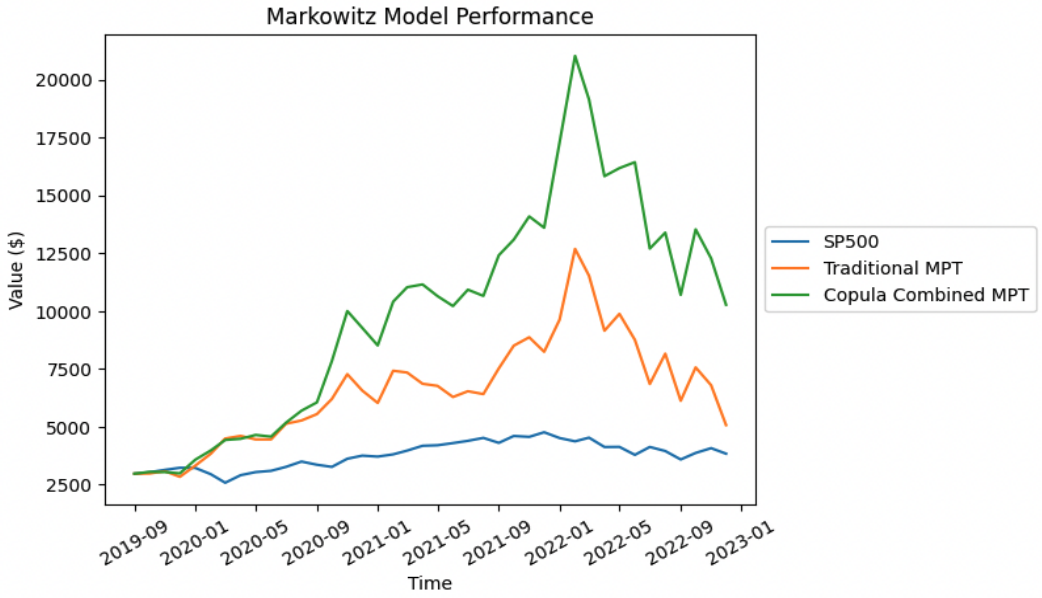

Figure 6: Copula Based MPT

As demonstrated above, the performance of the copula-based MPT exceeds that of the traditional MPT. However, it is important to note that the essence of this study is not to find a portfolio strategy that maximizes return, but it is to find a strategy that maximizes the risk-return trade-off. Although the copula-based MPT had a greater return in general, it

took a larger loss starting the start of 2022. Regardless of the poor performance of the market during this period, if we can find an alternative correlation matrix such as t-copula or another model that can predict the returns of the stocks, the MPT model will gain a meaningful improvement.

References

G. Alexander and A. Baptista. Economic implications of using a mean-var model for portfolio selection: A comparison with mean-variance analysis. *Journal of Economic Dynamics and Control*, 26(7-8):1159–1193, 2002. URL https://EconPapers.repec.org/RePEc:eee:dyncon:v:26:y:2002:i:7-8:p:1159-1193.

A. A. Ariyo, A. O. Adewumi, and C. K. Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, 2014. doi: 10.1109/UKSim.2014.67.

H. N. Bhandari, B. Rimal, N. R. Pokhrel, R. Rimal, K. R. Dahal, and R. K. Khatri. Predicting stock market index using lstm. *Machine Learning with Applications*, 9:100320, 2022. ISSN 2666-8270. doi: https://doi.org/10.1016/j.mlwa.2022.100320. URL https://www.sciencedirect.com/science/article/pii/S2666827022000378.

F. D. Freitas, A. F. De Souza, and A. R. de Almeida. Prediction-based portfolio optimization model using neural networks. *Neurocomputing*, 72(10):2155–2170, 2009. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2008.08.019. URL https://www.sciencedirect.com/science/article/pii/S092523120900040X. Lattice Computing and Natural Computing (JCIS 2007) / Neural Networks in Intelligent Systems Designn (ISDA 2007).

S. Gu, B. Kelly, and D. Xiu. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273, 02 2020.

R. Hamad, L. Yang, W. L. Woo, and B. Wei. Joint learning of temporal models to handle imbalanced data for human activity recognition. *Applied Sciences*, 10:5293, 07 2020. doi: 10.3390/app10155293.

S. Khan and H. Alghulaiakh. Arima model for accurate time series stocks forecasting. *International Journal of Advanced Computer Science and Applications*, 11(7), 2020. doi: 10.14569/IJACSA.2020.0110765. URL http://dx.doi.org/10.14569/IJACSA.2020.0110765.

T. Kim, J.-y. Shin, H. Kim, S. Kim, and J.-H. Heo. The use of large-scale climate indices in monthly reservoir inflow forecasting and its application on time series and artificial intelligence models. *Water*, 11:374, 02 2019. doi: 10.3390/w11020374.

M. Kumar and M. Thenmozhi. Forecasting stock index movement: A comparison of support vector machines and random forest. *SPGMI: Compustat Fundamentals (Topic)*, 2006.

Y. Ma, R. Han, and W. Wang. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, 165:113973, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113973. URL https://www.sciencedirect.com/science/article/pii/S0957417420307521.

A. Mahmoud and A. Mohammed. *A Survey on Deep Learning for Time-Series Forecasting*, pages 365–392. Springer International Publishing, Cham, 2021. ISBN 978-3-030-59338-4. doi: 10.1007/978-3-030-59338-4_19. URL https://doi.org/10.1007/978-3-030-59338-4_19.

H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2975974.

A. M. Rather. Lstm-based deep learning model for stock prediction and predictive optimization model. *EURO Journal on Decision Processes*, 9:100001, 2021. ISSN 2193-9438. doi: https://doi.org/10.1016/j.ejdp.2021.100001. URL https://www.sciencedirect.com/science/article/pii/S2193943821001175.

R. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 01 2000.

T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, 2013. doi: 10.1109/ICASSP.2013.6639347.

S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017. doi: 10.1109/ICACCI.2017.8126078.

J. Sen, A. Dutta, and S. Mehtab. Stock portfolio optimization using a deep learning LSTM model. In *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*. IEEE, oct 2021. doi: 10.1109/mysurucon52639.2021.9641662. URL https://doi.org/10.1109%2Fmysurucon52639.2021.9641662.

J. Shi, B. Tripp, E. Shea-Brown, S. Mihalas, and M. A. Buice. Mousenet: A biologically constrained convolutional neural network model for the mouse visual cortex. *PLOS Computational Biology*, 18:1–30, 09 2022. doi: 10.1371/journal.pcbi.1010427. URL https://doi.org/10.1371/journal.pcbi.1010427.

O. Ustun and R. Kasimbeyli. Combined forecasts in portfolio optimization: A generalized approach. *Computers Operations Research*, 39(4):805–819, 2012. ISSN 0305-0548. Special Issue on Operational Research in Risk Management.

H. Widiputra, A. Mailangkay, and E. Gautama. Multivariate cnn-lstm model for multiple parallel financial time-series prediction. *Complexity*, 2021, 10 2021. doi: 10.1155/2021/9903518.

J.-R. Yu, W.-J. Paul Chiou, W.-Y. Lee, and S.-J. Lin. Portfolio models with return forecasting and transaction costs. *International Review of Economics Finance*, 66:118–130, 2020. ISSN 1059-0560. doi: https://doi.org/10.1016/j.iref.2019.11.002. URL https://www.sciencedirect.com/science/article/pii/S1059056018309614.

R. Zhang. Lstm-based stock prediction modeling and analysis. In *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)*, pages 2537–2542. Atlantis Press, 2022. ISBN 978-94-6239-554-1. doi: 10.2991/aebmr.k.220307.414. URL https://doi.org/10.2991/aebmr.k.220307.414.

Z. Zhang, S. Zohren, and S. Roberts. Deep learning for portfolio optimization. *The Journal of Financial Data Science*, 2(4):8–20, aug 2020. doi: 10.3905/jfds.2020.1.042. URL https://doi.org/10.3905%2Fjfds.2020.1.042.

7 Appendix

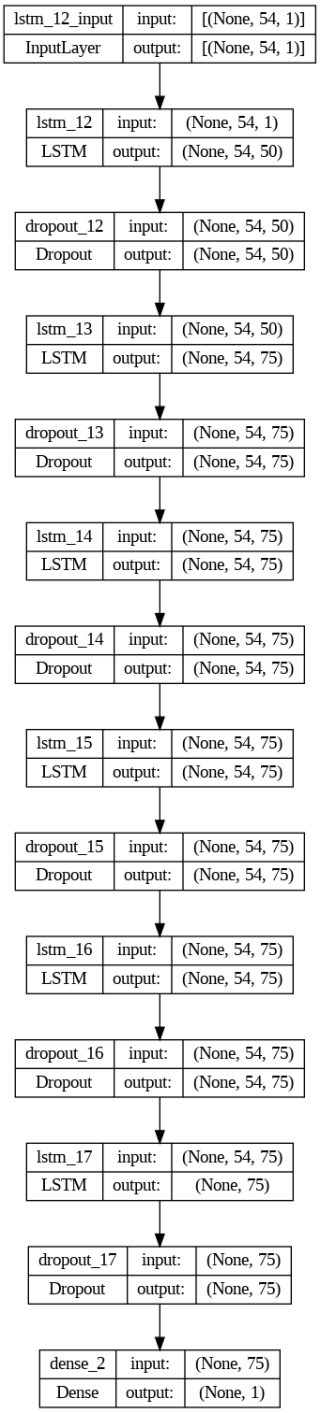

Figure 7: The schematic diagram of an LSTM

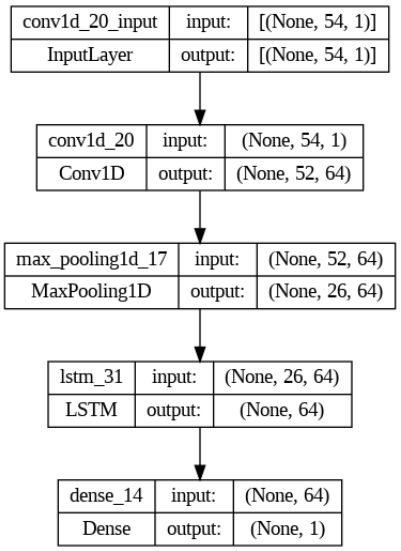

Figure 8: The schematic diagram of a CNN-LSTM