CARMA 2024

6th Int. Conf. on Advanced Research Methods and Analytics
Universitat Politècnica de València, València, 2024
DOI: https://doi.org/10.4995/CARMA2024.2024.17554

Unlocking the Potential of Machine Learning in Portfolio Selection: A Hybrid Approach with Genetic Optimization

Chaher Alzaman

Department of Supply Chain and Business Technology Management, John Molson School of Business, Concordia University, Montreal, Quebec, H3H 0A1, Canada.

How to cite: Alzaman, C. 2024. Contemporary issues in Financial Technology: the role of the Internet. In: 6th International Conference on Advanced Research Methods and Analytics (CARMA 2024). Valencia, 26-28 June 2024. https://doi.org/10.4995/CARMA2024.2024.17554

Abstract

In the field of financial market predictions, machine learning has been widely used to identify patterns and gain valuable insights. However, for success in portfolio selection, it is crucial to optimize factors that impact accuracy. This study focuses on combining machine learning and optimization to enhance stock selection and prediction capabilities, thereby addressing a critical challenge faced by investors and traders. The work starts with hyper-parameter optimization and utilizes three different machine learning algorithms: XGBoost, LSTM, and Deep RankNet. These algorithms were chosen for their proven performance in handling complex financial data and capturing nonlinear relationships. Our findings show a 40% improvement in results through the use of a genetic-based optimization technique, as well as a promising daily average return of 0.47% through a novel feature engineering approach. The study provides a framework for optimizing and learning in financial portfolio selection, with promising results for medium and small-sized traders who often face resource constraints in developing sophisticated trading strategies. The proposed approach offers a scalable and adaptable solution that can be tailored to different market conditions and investment objectives.

Keywords: *Artificial Intelligence; Machine Learning; Optimization; Financial Markets; Predictive Analytics.*

1. Introduction

Machine learning has gained widespread attention in the financial market as a tool for predicting stock prices, foreign exchange rates and other market trends. With its ability to analyze large amounts of data, machine learning algorithms can provide more accurate predictions compared to traditional statistical methods. Shah (2007) highlights two main approaches in stock

prediction: Fundamental Analysis, where analysis is based on a company's financial characteristics (such as past performance, assets, earnings, etc.), and Technical Analysis, where patterns in past stock prices are studied. Despite the efficient market hypothesis (Jensen, 1978) stating that stock prices cannot be predicted and the random-walk hypothesis (Malkiel, 1973) suggesting stock prices only depend on future information and not on history, research by Basak et al. (2019), Chen et al. (2020), and others argue that some elements of stock behavior are predictable.

This work employs LSTM, XGBoost, and Deep RankNet. To set a background, two classes of methods have been prominent in the literature (Machine Learning applications in financial markets): Artificial Neural Networks (ANN) and Ensemble tree-based algorithms. ANN is at the heart of Deep Learning, which in turn is a subset of Machine Learning (ML) geared toward more complex systems (e.g., big data). LSTM (long short-term memory) is an artificial recurrent neural network tailored to sequential data, such as closing prices of financial assets. The XGBoost (XGB) is an ensemble decision tree-based algorithm that is quite popular in financial market predictions. Basak et al. (2020) and Chen et al. (2021) confirm the effectiveness of XGB. Our newest method, Deep RankNet, is a novel approach in predicting financial assets, first introduced by Burges et al. (2005). The concept behind Deep RankNet involves using deep learning to rank and match objects. Li and Tan (2021) have applied Deep RankNet to ranking and forming trading portfolios in the field of financial market predictions. This work supplements the above with Hyperparameter optimization, which involves adjusting critical parameters in machine learning. The user sets these parameters prior to training. But for complex and intricate systems such as financial systems, proper tuning is crucial as financial assets are notorious for being noisy and sensitive to small variations in the input.

2. Literature Review

Many studies in the field of predictive analytics in financial markets compare the effectiveness and performance of various Machine Learning (ML) techniques. These studies evaluate different ML algorithms and choose the best one(s) for financial market prediction. Some examples of these studies include works by (Basak et al., 2019, Kumar et al. 2018, Patel et al. 2019, Ismail et al. 2020, Usmani et al. 2016, Nelson et al. 2017, Shen et al. 2012, Singh and Srivastava 2017, and Vijh et al. 2020). The works mainly examine different strategies for predicting financial market outcomes. On the other hand, there are studies that aim to improve the prediction performance of one or more ML techniques, such as (Porshnev et al. 2013, Wang et al. 2018, Kim et al. 2020, Akita et al. 2016, Reddy 2018 and Vazirani et al. 2020).

In the field of Portfolio Selection, there have been several studies that apply different techniques. Chen et al. (2020) use XGBoost with multi-features, including technical indicators, for selection and hyperparameter optimization in the Shanghai Stock Exchange. Liu and Yeh (2017) employ

neural networks to predict the behavior of American stocks, but do not perform feature selection or hyperparameter optimization. Paiva et al. (2019) suggest using a Support Vector Machine (SVM) for stock predictions in Sao Paulo and conduct hyperparameter optimization. Long et al. (2019) utilize a multi-filter neural network for feature extraction and price movement prediction of financial time series data, using 1-minute stock price frequency and convolutional and recurrent neurons. It is important to note that small and medium investors may not have access to such detailed data, especially in smaller trading markets.

One of this work's main contributions is the explicit use of a genetic-based algorithm to tune the machine learning hyper-parameters. Dessain (2022) classifies predicting returns' performance metrics, within the context of machine learning, into error-based, accuracy-based, and investment-based metrics. The first metric is associated with computing prediction errors. The second metric is the measure of returns based on the algorithm's prediction accuracy. The last is concerned with result-based metrics and risk-adjusted return-based metrics. This work employs all three categories and focuses on risk-adjusted return-based metrics.

3. Methods

3.1. LSTM

LSTM is a form of a recurrent neural network. Figure 1 illustrates a schematic for LSTM, where multiple gates encompass activation functions. It closely follows the work of Fischer and Krauss (2018). LSTM can be thought of as a system of cells where information gets added or removed to the cell state by the use of gates depending on its importance (important information kept, irrelevant information discarded).

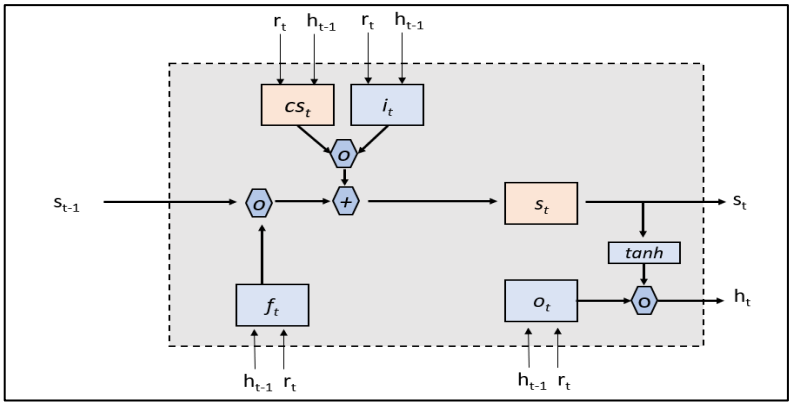

Figure 1: LSTM Architecture of memory cell (f, forget; i, input cell; cs, candidate state value; s, cell state; o, output cell)

We utilize 42-time steps (t), where at every t the input is posed as xt and an output ht-1 of the memory cell at the previous t (t-1). As per the work of Fischer and Krauss (2018), we utilize the stock return as an input parameter based on the given stock closing P (Pt) for a specific period t.

$$R_t = \frac{P_t}{P_{t-1}} - 1 \tag{1}$$

In essence, the gates act as filters (see figure 1). The forget gate dictates which information to remove from the memory cell state. While the input gate ushers which information to add to the memory cell state. Finally, the output gate directs which information from the memory cell state to utilize as output. The equations below characterize the LSTM procedure in a vector form. Candidate state shall be represented as cs, while state cell is indicated by s alone. Input and output values are designated as i and o.

- rt is an input vector at time step (t)
- The weight matrices are Wf,r, Wf,h, Wcs,r, Wcs,h, Win,h, Wo,r, and Wo,h.
- The bias vectors are bf, bcs, bin, and bo.
- Activation function vectors are ft, it, and ot for the three gates respectively.
- The output vector for the LSTM layer is ht.

The cell states and output are updated using the following procedure. First, LSTM needs to decide which information to discard. This is done using the activation function (Eqn. 2).

$$f_t = sigmoid(W_{f,r}r_t + W_{f,h}h_{t-1} + b_f) \tag{2}$$

We have ran two instances of a sigmoid and linear activation function, and found the first to be superior. Second, the LSTM layer decides which information to be added to the cell states in two parts. First, the candidate state value is computed (Eqn. 3). Second, the activation value of the input gate is computed (Eqn. 4)

$$cs_t = tanh(W_{cs,r}r_t + W_{cs,h}h_{t-1} + b_{cs}) \tag{3}$$

$$i_t = sigmoid(W_{i,r}r_t + W_{i,h}h_{t-1} + b_i) \tag{4}$$

In the third step, the new cell states s, t, are calculated based on the results of the previous two steps denoting the Hadamard (elementwise) product. Then, the Hadamard product is used to arrive at the new cell state value (Eqn. 5).

$$s_t = f_t \cdot s_{t-1} + i_t \cdot cs_t \tag{5}$$

Finally, the output of the memory cell is computed using equations 6 and 7.

$$o_t = sigmoid(W_{o,r} r_t + W_{o,h} h_{t-1} + b_o) \tag{6}$$

$$h_t = o_t \cdot \tanh(s_t) \tag{7}$$

The data input for the LSTM and all three algorithms constitutes thirty-two training and two holding days as per the work of Li and Tan (2021). However, the input for all three algorithms is amalgamated to enhance the learning capabilities of both algorithms. This is a primary contribution of this work. In effect, for each stock entry, six lags are created. So, the input matrix is the vector of S ($\forall s \in S \times NL$) stocks where NL stands for the number of lags applied. As shown in the exemplary matrix below, for eleven-day training data, we construct six lags where each lag contains six days vector.

$$\begin{bmatrix} R_1 & \cdots & \ldots & R_6 \\ R_2 & \ldots & \ldots & R_7 \\ \vdots & \ddots & \ddots & \vdots \\ R_5 & \cdots & \ldots & R_{10} \\ R_6 & \ldots & \ldots & R_{11} \end{bmatrix}$$

We have contested different NL values of {2, 3, 5, 6, 7, 8, 9} and found six to be optimal. Higher NL's renders a higher model complexity and a deterioration in performance. Second, we study different periods (size and boundaries) and subperiods as should be indicated in the results' figures in the next sections. Figure 2 exhibits the procedural code for the LSTM algorithm used in this work. A set of 100 TSE stocks S {s1, s2, …., s100} is used to compute daily returns (eqn. 1). Then the returns are ranked with respect to the median (Fischer & Krauss 2018), MS, of the set S. Stocks with higher returns than MS are assigned to Class 1, lower stocks are assigned to Class 0. The values of the hyperparameters are pre-set to the output optimal values of the Genetic Algorithm (to be discussed at the end of this section). Ultimately, the LSTM will handle the input matrix and output value of one or zeros depending on the class.

→Compute excess return for all ***Stocks***
→Compute the median of each stock
→Assign top 50% stocks that are higher than the median to *Class 1*
→Assign lowest 50% stocks a ranking label of *Class 0*
→Construct Input Feature Matrix of (N_{stocks} * NL)× N_{days} (number of stocks, NL, number of days)
→Optimize LSTM hyperparameters
→Set output of Class 1 stocks to 1
→ Set output of Class 0 stocks to 0
→Execute equation 2 to 7 (deep learning)

Figure 2: LSTM Technique

3.2. Deep RankNet

Deep RankNet has been designed to learn the relative performance of documents/queries in a pairwise manner. It was pioneered by Burges et al. (2005). In their work, they use a probabilistic cost function, which utilizes a pair of sample objects to instigate and learn how to rank documents/objects. The cost function effectively seeks to minimize the number of pairing instances needed to correctly order a set of items. Given the challenges of financial asset predictions, Li and Tan (2021) bring forward a Deep RankNet to rank the performance of financial assets and ultimately bundle high-performing stocks into trading portfolios. The model instituted in this work will implement the framework introduced by Li and Tan (2021). However, we integrate novel elements of subperiods analysis. To start, an excess return is computed for a given stock relative to the composite index (market index). Here we contest 100 stocks all traded in the TSE. The equations for the return rate of a given stock, Rt,s, return of the index Rt,I, and excess return, ER (Li and Tan, 2021) are presented below.

$$R_t^s = \frac{P_t^s}{P_{t-1}^s} - 1 \tag{8}$$

$$R_t^I = \frac{P_t^I}{P_{t-1}^I} - 1 \tag{9}$$

$$ER_{t,m}^S = R_{t,m}^s - R_{t,m}^I \tag{10}$$

R = Return made by a given stock (s□S); P = Closing Price a stock (or index i); t = trading day; ER= Excess return of stock s relative to index i, for holding period m. Figure 3 illustrates the Deep RankNet procedure used in this work. The procedure follows closely the work of Li and Tan (2021).

→Compute excess return for all N_{stocks}
→Rank Stocks' returns from highest to lowest
→Assign top 50% stocks a ranking label of *1_Class 1*. Assign lowest 50% stocks a ranking label of *0_Class 2*
→Feed the *ER's* into a neural deep network as input
→Input Matrix is $N_{stocks} \times N_{days}$ (number of stocks, number of days)
→→Each stock entry includes fixed number of days (N_{days}), past *ER's*
→Initiate Neural Network outputs
→→*Score 1* for *Class 1* data
→→*Score 2* for *Class 2* data
→→→Subtract *score 2* from *score 1* to produce *Dif*
→→→Input *Dif* into a Sigmoid function
→Apply loss cross entropy function
→Apply gradient descent
→Update Neural Network parameters
→Output neural network scoring model

Figure 3: Deep RankNet procedure

3.3. XGBOOST (XGB)

XGBoost (XGB) stands for eXtreme Gradient Boosting. It was introduced by Chen (2016). The method achieves high accuracy and exceptionally fast speed due to its low computational complexity. The idea behind the method is to combine a penalty term and a loss function term. In turn, the method aims at obtaining high accuracy solution by minimizing the penalty term while preventing over-fitting via the reduction of the model's variance (Chen and Guestrin, 2016). The XGB method uses K additive function to predict output:

$$\widehat{y_i} = \varphi(x_i) = \sum_{k=1}^{K} f_k(x_k)\,,\ f_k \epsilon F \tag{11}$$

Where F is the space of regression trees and fk corresponds to each tree structure q and leaf weight w. The space of the regression tree can be written as:

$$F = \left\{w_{q(X)}\right\} for\ q: \mathbb{R}^m \to T, w \in \mathbb{R}^T \tag{12}$$

Where T is the number of leaves in the tree for *m* features. The objective function in the ensemble tree model is:

$$L(\varphi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{13}$$

Where l is a differentiable convex loss function and omega penalizes the complexity. Given the limitation of ensemble models as they cannot be optimized using traditional methods (in Euclidean space), the XGB is trained in an additive manner.

$$L(\varphi) = \sum_i l(\hat{y}_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{14}$$

Equation 14 integrates a greedy ft function, which improve the original model (eq. 13) significantly (Chen and Guestrin, 2016). This work optimizes the value of Gama, γ, in the $\Omega(f_t)$ term:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda\|w\|^2 \tag{15}$$

The work optimizes the values of γ, T (see eqn. 12), and F (eqn. 11 and 12). Figure 4 depicts the pseudo code of the XGB algorithm.

→Compute excess return for all Stocks

→Compute the median of each stock

→Assign top 50% stocks that are higher than the median to Class 1

→Assign lowest 50% stocks a ranking label of Class 0

→Construct Input Feature Matrix of (Nstocks*

Figure 4: XGBoost Procedure

3.4. Genetic Algorithm

Genetic Algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. Pertaining to this work, the neural network's hyperparameters are numerous and quite difficult to tune. We start by populating an N random instances matrix for M-vector elements. We shall label each row as a chromosome, where each chromosome consists of a set of values for the hyperparameters in the question. We run the neural network for each chromosome and produce a unique prediction error. Then, we sort the chromosomes from lowest error value to highest. Next, the top half of our population table/matrix is chosen for mating (i.e., natural selection). Mating produces offspring that will populate the top of the table while the parents (top half of the previous iterate) will populate the bottom half. This way, if the parents turn out to be more effective than their offspring, they are not lost in the process (Goldberg, 1989; Holland, 1975). Then comes mutation where in each generation, we introduce 4% new random chromosomes to assure that a broader exploration of the solution space is warranted.

4. Results

The following subsections aim to provide a comprehensive analysis of the performance of the selected machine learning algorithms. This includes exploring the impact of hyperparameter

optimization on the algorithms' results, and evaluating the performance of different machine learning techniques.

4.1. Hyper-parameters

To demonstrate the impact of combining learning with optimization, we first present a random grid search. Our initial focus is solely on stocks traded on the Toronto Stock Exchange (TSE), for which we use 100 TSE stocks (closing prices obtained via Yahoo Finance's Python API) during the study period. The study requires complete data sets throughout all periods, thus several preprocessing steps were taken to eliminate financial assets with missing data, unresponsive updates (due to the API used), and inconsistencies. The hyper-parameters, to be optimized in this study, include the training ratio, batch size, number of neurons, stopping criterion, number of lags, number of hidden layers, and number of days in a time period. The batch size is a key factor that balances the speed of training with the quality of training. The number of neurons can add complexity to the model but may also result in overfitting. The stopping criterion sets the maximum number of non-improvements allowed in the solution before termination to avoid missing the global minimum. The number of hidden layers and lags also impact the predictions. The size of the time period in the training data can also have an impact on the results, with longer periods favored in the literature but shorter periods giving more emphasis on recent inputs. Figure 5 illustrates the correlation between these factors and DFs, which is the prediction error.

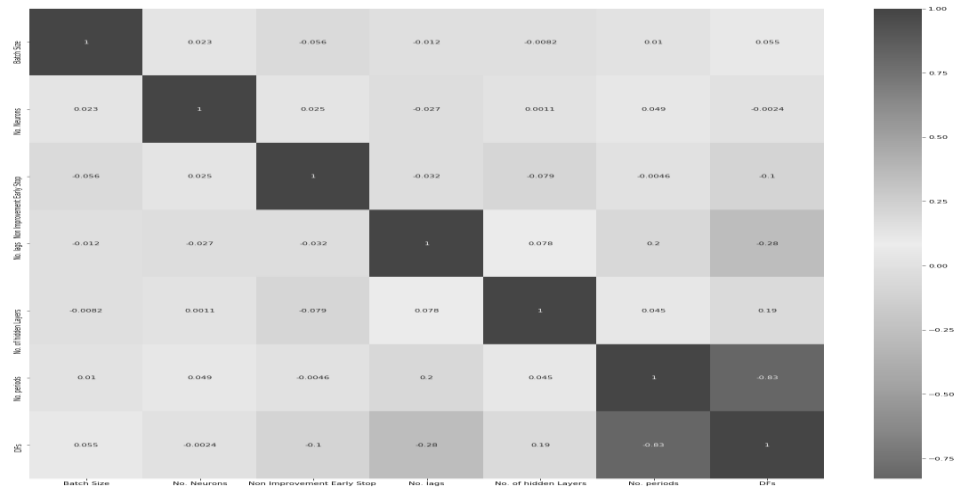

Figure 5: heat map of the correlations of the hyperparameter (FAP, DFs is the prediction error)

We utilize the genetic algorithm (GA) introduced in section 3 to optimize the hyperparameters (see figure 5). The training ratio of 80 to 20: 80% training and 20% turn to perform better and

is going to be used for the algorithms. Overall, the genetic-based algorithm does improve prediction accuracy. The hyperparameter optimization brings optimal value for a batch size of six, number of neurons of 22, and one hidden layer. For the XGB (XGBoost), the optimal value of gamma (eqn. 15) is 0.02. While the optimal learning rate occurs at 0.01. The optimal value for T (eqn. 11) is 300 and the optimal value of F (eqn. 11 & 12)is 8. Overall, we see 40% improvement in results with the use of the genetic technique.

4.2. Learning Algorithms

In this work, we bring add a novel element of feature engineering by aggregating data through the use of lags. We have found that this is especially beneficial for deep learning, as larger input data improves the training process. We employ period lags to manipulate the input, resulting in a higher quantity and quality of input data. Our preliminary research indicates that this feature engineering approach improves the accuracy of both XGB and LSTM algorithms.

The focus of this study is on portfolios with k=5 stocks. Three non-overlapping periods from 2019-01-01 to 2021-08-09 were analyzed. In the first two periods (as shown in Table 1), both XGB and LSTM outperformed the market in terms of mean return. Each period consisted of 200 days, approximately one calendar year. Notably, both techniques also outperformed the market during a negative revenue period in the second period. Despite the high variability in performance between the three techniques due to market fluctuations, all techniques performed better than the market during the initial wave of the COVID-19 pandemic, which is in line with previous studies (Bogomolov, 2013; Do and Faff, 2010; Huck, 2010) that have shown the effectiveness of pair trading strategies in periods of high turmoil. Looking across all three periods, we can see an apparent advantage of LSTM over the two other algorithms with an average daily return of 0.47% (compared to 0.29% for XGB). Alternatively, the Deep Rank performs poorly, with a negative daily return average. If we incorporate a 0.16% transaction cost as recommended in the work of Li and Tan (2021), we still see a marginal profit of 0.31% for the LSTM. This compares well with the work of Krauss et al. (2017). According to their analysis, they explain the very disappointing result from 2010 to 2015 as caused by an increase in public availability of powerful machine learning algorithms. If we assume the trend persists, the results, presented in the work, are quite significant since they still show overall positive daily return averages that are higher than the market.

Table 1: Three recent period comparisons between algorithms (200 days period)

	XGB	**LSTM**	**Deep Rank**	**Market**
Period from 2019-01-17 to 2020-01-06				
Mean	0.003066	0.003994	0.000377	0.000605
Standard Deviation	0.023545	0.033864	0.045080	0.006544
Sharpe ratio	1.455803	1.318577	0.093580	1.033376
Sortino ratio	2.559949	3.219885	0.208390	1.438714
Period from 2019-11-04 to 2020-10-21				
Mean	0.003454	0.002814	-0.004810	-0.000049
Standard Deviation	0.050973	0.054202	0.048303	0.027516
Sharpe ratio	0.757664	0.580408	-1.113287	-0.019919
Sortino ratio	1.365378	0.902510	-1.939949	-0.021003
Period from 2020-08-20 to 2021-08-09				
Mean	0.002358	0.007413	0.003212	0.002232
Standard Deviation	0.029439	0.051946	0.046531	0.009321
Sharpe ratio	0.895459	1.595566	0.771838	2.677102
Sortino ratio	1.760057	3.729138	2.044510	3.867835

Looking at the historic tail risk of the portfolios, based on the historic 1% and 5% VAR (value at risk), we measure the extent of potential extreme financial losses. These figures are not reported due to space limitation but can be provided upon request. Effectively, for all three periods (average), the 1%-VAR is higher for the LSTM standing at 9.7% while it is 8.0% for XGB illustrating a slight increase of risk while the 5%-VAR is quite similar between the two. The Sharpe ratio, which expresses the excess return per unit of risk, quantified in standard deviations, for LSTM is 0.13 points higher than for XGB. The Sortino ratio, which scales the returns by their downside deviation (Krauss et al., 2017) shows an upper hand for the LSTM with a ratio of 2.617 compared to 1.895 for the XGB.

Looking at figure 6, we can see the overall performance of the LSTM for three consecutive years, each year consisting of roughly 250 trading days. The Y-axis represents the daily return (not percentage), while the X-axis notes the period. In the figure, we observe a higher overall return in the LSTM model than the XGB. This correlates with the findings in Table 1, however, LSTM does exhibit higher variability than XGB. This is in line to the work of Chen et al. (2021).

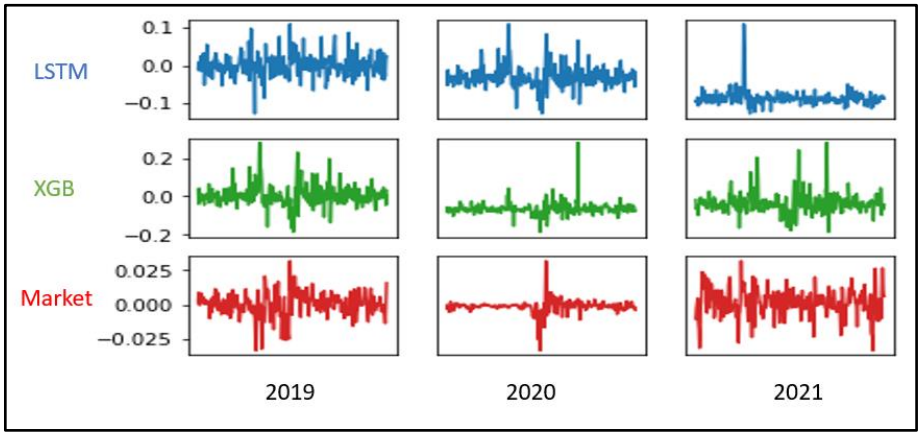

Figure 6: Average daily returns corresponding to year for XGB and LSTM

5. Conclusion and Recommendations

In this work, we utilize deep learning techniques to analyze financial data and make predictions for stock prices. Additionally, we employ ensemble decision trees to further enhance the accuracy of our predictions. The combination of these models is optimized through parameter tuning to achieve the best results for portfolio selection in trading. The ultimate goal is to optimize the holding period for maximum return on investment. The work studies a wide range of hyperparameters that affect the performance of the algorithms. We've taken the time to explicitly discuss many of the parameters in this work: Training ratio, Batchsize, Stopping criterion, period size, and others. The results indicate significant gains reaped by the optimizing the hyperparameters where the genetic algorithm brings 40% improvement compare to traditional random-grid approaches. To the best of our knowledge, this is the first work to coalesce Deep RankNet, LSTM, and a genetic-based algorithm.

Future research could focus on three key areas to further advance our understanding of algorithmic portfolio construction. Firstly, exploring different ranking rules for selecting stocks into portfolios would provide insights into the impact of different methods on portfolio performance. Secondly, more explanatory work is needed to understand why certain algorithms perform better than others in certain situations. This could involve detailed analysis of the underlying data patterns and market dynamics that drive algorithm performance. Lastly, expanding the scope of the research beyond North America to other financial markets would provide a more comprehensive understanding of algorithmic portfolio construction. This could involve testing the same algorithms on data from different financial markets and evaluating the performance in terms of return and risk characteristics.

Reference

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K.: Deep learning for stock prediction using numerical and textual information. IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), 1-6 (2016).

Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R.: Predicting the direction of stock market prices using tree-based classifiers. The North American Journal of Economics and Finance, 47, 552-567 (2019).

Bogomolov, T.: Pairs trading based on statistical variability of the spread process. Quantitative Finance, 13(9), 1411–1430 (2013).

Burges, C., S. Tal., R. Erin., L. Ari., D. Matt., H. Nicole, and H. Greg.: Learning to Rank Using Gradient Descent. Proceedings of the 22nd International Conference on Machine Learning, 89–96 (2005).

Chen, T., & Guestrin, C.: Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm SIGKDD international conference on knowledge discovery and data mining, 785-794 (2016).

Chen, W., Zhang, H., Mehlawat, M. K., & Jia, L.: Mean–variance portfolio optimization using machine learning-based stock price prediction. Applied Soft Computing, 100, 106943 (2021).

Coqueret, G.: Persistence in factor-based supervised learning models. The Journal of Finance and Data Science, 8, 12-34 (2022).

Dessain, J.: Machine learning models predicting returns: Why most popular performance metrics are misleading and proposal for an efficient metric. Expert Systems with Applications, 199, 116970 (2022).

Do, B. , & Faff, R.: Does simple pairs trading still work? Financial Analysts Journal, 66(4), 83–95 (2010).

Du, J.: Mean–variance portfolio optimization with deep learning based-forecasts for cointegrated stocks. Expert Systems with Applications, 201, 117005 (2022).

Fischer, T., & Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669 (2018).

Goldberg, D.: Genetic algorithm in search, optimization and machine learning. Addison-Wesley, Reading, MA (1989).

Holland, J. H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975).

Huck, N.: Pairs selection and outranking: An application to the S&P 100 index. European Journal of Operational Research, 196(2), 819–825 (2009).

Huck, N.: Pairs trading and outranking: The multi-step-ahead forecasting case. European Journal of Operational Research, 207(3), 1702–1716 (2010).

Huck, N.: Pairs trading: Does volatility timing matter? Applied Economics, 47(57), 6239–6256 (2015).

Ismail, M. S., Noorani, M. S. M., Ismail, M., Razak, F. A., & Alias, M. A.: Predicting next day direction of stock price movement using machine learning methods with persistent

homology: Evidence from Kuala Lumpur Stock Exchange. Applied Soft Computing, 106422 (2020).

Jensen, M.: Some anomalous evidence regarding market efficiency. J. Financ. Econ., 6(2–3), 95-101 (1978).

Kim, S., Ku, S., Chang, W., & Song, J. W.: Predicting the Direction of US Stock Prices Using Effective Transfer Entropy and Machine Learning Techniques. IEEE Access, 8, 111660-111682 (2020).

Krauss, C., Do, X. A., & Huck, N.: Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), 689-702 (2017).

Kumar, I., Dogra, K., Utreja, C., & Yadav, P.: A comparative study of supervised machine learning algorithms for stock market trend prediction. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 1003-1007 (2018).

Li, Y., & Tan, Z.: Stock Portfolio Selection with Deep RankNet. The Journal of Financial Data Science, 3(3), 108-120 (2021).

Malkiel, B. G.: A Random Walk Down Wall Street. Norton, New York (1973).

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A.: Stock market's price movement prediction with LSTM neural networks. In 2017 International joint conference on neural networks (IJCNN), 1419-1426 (2017).

Patel, J., Shah, S., Thakkar, P., & Kotecha, K.: Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert systems with applications, 42(1), 259-268 (2015).

Paiva, F. D., Cardoso, R. T. N., Hanaoka, G. P., & Duarte, W. M.: Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. Expert Systems with Applications, 115, 635-655 (2019).

Porshnev, A., Redkin, I., & Shevchenko, A.: Machine learning in the prediction of stock market indicators based on historical data and data from twitter sentiment analysis. In 2013 IEEE 13th International Conference on Data Mining Workshops (pp. 440-444) (2013).

Reddy, V. K. S.: Stock market prediction using machine learning. International Research Journal of Engineering and Technology, 5(10) (2018).

Ren, R., Wu, D. D., & Liu, T.: Forecasting stock market movement direction using sentiment analysis and support vector machine. IEEE Systems Journal, 13(1), 760-770 (2018).

Shah, V. H.: Machine learning techniques for stock prediction. Foundations of Machine Learning, Spring, 1(1), 6-12 (2007).

Shen, S., Jiang, H., & Zhang, T.: Stock market forecasting using machine learning algorithms. Department of Electrical Engineering, Stanford University, Stanford, CA, 1-5 (2012).

Singh, R., & Srivastava, S.: Stock prediction using deep learning. Multimedia Tools and Applications, 76(18), 18569-18584 (2017).

Sharma, M., & Shekhawat, H. S.: Portfolio optimization and return prediction by integrating modified deep belief network and recurrent neural network. Knowledge-Based Systems, 109024 (2022).

Shi, S., Li, J., Li, G., Pan, P., Chen, Q., & Sun, Q.: GPM: A graph convolutional network based reinforcement learning framework for portfolio management. Neurocomputing, 498, 14-27 (2022).

Tim, J. M.: Artificial Intelligence–A System Approach. Computer Science Series, Infinity Science Press, 498 (2008).

Usmani, M., Adil, S. H., Raza, K., & Ali, S. S. A.: Stock market prediction using machine learning techniques. In 2016 3rd international conference on computer and information sciences (ICCOINS), 322-327 (2016).

Vazirani, S., Sharma, A., & Sharma, P.: Analysis of various machine learning algorithm and hybrid model for stock market prediction using python. In 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 203-207 (2020).

Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A.: Stock Closing Price Prediction using Machine Learning Techniques. Procedia Computer Science, 167, 599-606 (2020).

Wang, G., Gunasekaran, A., & Ngai, E. W.: Distribution network design with big data: model and analysis. Annals of Operations Research, 270(1-2), 539-551 (2018).

Wu, D., Wang, X., & Wu, S.: A Hybrid Framework Based on Extreme Learning Machine, Discrete Wavelet Transform, and Autoencoder with Feature Penalty for Stock Prediction. Expert Systems with Applications, 118006 (2022).

Yadav, A., Jha, C. K., & Sharan, A.: Optimizing LSTM for time series prediction in Indian stock market. Procedia Computer Science, 167, 2091-2100 (2020).

Zhao, D., Bai, L., Fang, Y., & Wang, S.: Multi-period portfolio selection with investor views based on scenario tree. Applied Mathematics and Computation, 418, 126813 (2022).