DOI: 10.15439/2024F885 ISSN 2300-5963 ACSIS, Vol. 39

Forecasting Stock Trends with Feedforward Neural Networks

Marcin Traskowski
University of Warsaw
Warsaw, Poland
Email: traskowski.marcin@gmail.com

Eyad Kannout
University of Warsaw
Warsaw, Poland
Email: eyad.kannout@mimuw.edu.pl

***Abstract*—Stock market prediction stands as a complex and crucial task, pivotal for enhancing the overall stability and efficiency of financial markets by offering essential insights into market movements and trends. In this study, we introduce a simple yet potent model based on feedforward neural networks to tackle this challenge effectively. Our approach leverages advancements in machine learning and deep learning to analyze large datasets of financial statements, demonstrating promising results in forecasting stock trends.**

Index Terms—Stock Market Prediction, Neural Networks, Deep Learning, Forecasting, Classification

I. Introduction

FORECASTING stock trends has long been an intriguing and challenging problem for researchers and enthusiasts in the fields of finance and data science. Accurate predictions can lead to significant financial gains and help investors make informed decisions. The factors and sources of information to consider are numerous and diverse, making it very challenging to foresee future stock market behavior accurately [1]. It is clear that precise prediction of stock prices is elusive. Fama et al. [2] introduced the efficient market hypothesis, which asserts that an asset's current price always reflects all available prior information. Additionally, the random walk hypothesis, introduced by Burton [3], proposes that a stock's price movements are independent of its past, implying that tomorrow's price will rely exclusively on tomorrow's information, irrespective of today's price. Together, these hypotheses suggest that accurately predicting stock prices is impossible. Nonetheless, extensive research has been conducted to address this issue, proposing a range of methodologies across multiple disciplines, including economics, statistics, physics, and computer science [4].

Traditional methods of stock trend analysis often rely on statistical models, such as ARIMA [5], which may not effectively capture the complex, non-linear, and often unstable patterns present in financial data. In recent years, advancements in machine learning and deep learning have opened new avenues for analyzing and predicting stock trends [6]. Neural networks, in particular, have shown promise due to their ability to learn and generalize from large datasets. This paper presents a simple yet effective neural network approach to predicting stock trends, developed as part of the FedCSIS 2024 Data Mining Challenge [1] [7] .

[1]https://knowledgepit.ai/fedcsis-2024-challenge/

The paper is organized as follows: the next part reviews related work in stock predictions, followed by a description of the competition data and preprocessing methods. Next, we detail the model architecture, hyperparameters and evaluation metrics. Subsequently, we present the experimental results. The last part is reserved for final conclusions and observations.

II. Related Works

Recent advancements in machine learning and deep learning have significantly enhanced the ability to predict stock market trends. Numerous studies have explored different approaches and models, showcasing varying degrees of success and innovation.

Random Forest, an ensemble learning method, has been widely used for its robustness and accuracy in stock market prediction. By constructing multiple decision trees and aggregating their results, Random Forest reduces overfitting and improves generalization. Its ability to handle large datasets with high dimensionality makes it particularly effective for financial market predictions [8].

Other research endeavors have concentrated on employing support vector machines (SVMs) to improve the accuracy of stock market predictions through categorization of examples. SVM models represent examples as points in a multidimensional space, with the objective of maximizing the separation between different categories. New examples are then classified based on the category they are most likely to belong to [9]. Liu et al. [10] developed a model using the RBF-SVM algorithm to enhance stock price prediction, accurately assess short-term stock price movements, and provide more reliable guidance for stock market analysis and investor decision-making.

Another prominent method involves using Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) due to their ability to handle sequential data [11]. Research has shown that LSTMs can outperform traditional machine learning models in predicting stock price movements by capturing temporal dependencies in financial data [12].

The application of Convolutional Neural Networks (CNNs) to extract spatial features from stock market data has also shown promise. Research has utilized hybrid models combining CNNs and LSTMs to predict stock prices, showing improved performance over standalone models [13].

Attention mechanisms, particularly through Transformer models, have demonstrated significant potential in stock trend prediction. By effectively handling long-term dependencies and integrating both technical indicators and sentiment analysis, Transformer-based models like StockFormer [14] have shown superior predictive capabilities compared to traditional RNN-based models. Furthermore, stock market prices are significantly influenced by the sentiments of stakeholders, which can be assessed through the analysis of news, social media data, and other indicators. Kumar et al. [15] explored the correlation between the sentiment polarity of news and a company's stock price. They employed the SVM-LSTM-GRU Composite Model to predict a company's stock price based on news related to that company.

The research literature has documented numerous efforts to develop recommendation systems [16], [17] that advise users on whether to buy or sell stocks [18]. Additionally, Association Rule Mining (ARM) is widely used to create recommender systems [19]. In stock predictions, several systems exist for monitoring and predicting stock prices, but they typically focus on individual stocks and do not account for the inter-relationships between stocks or their connections with the stock market index. Paranjape-Voditel et al. [20] employed various ARM techniques—such as fuzzy ARM, weighted fuzzy ARM, ARM with time lags, fuzzy ARM with time lags, and weighted fuzzy ARM with time lags—to predict relationships between stocks. This approach forms the basis for portfolio management and provides recommendations for mutual funds.

These studies collectively indicate that leveraging advanced deep learning architectures, ensemble methods, and attention mechanisms can significantly enhance the predictive accuracy of stock market models.

III. Datasets

The competition included training and test sets, consisting of 8,000 and 2,000 examples, respectively. Each example represents a financial statement announcement for one of the chosen 300 companies. Each record consists of 119 elements. The first column is a categorical value that represents the company's sector. The next 58 columns contain values for key financial indicators. The following 58 columns represent the 1-year (absolute) change for each of these previous financial indicators. The last two columns are targets:

1) The target column "Class" can have three possible values:
   - -1: "sell" (do not invest)
   - 0: "hold"
   - 1: "buy (invest)
2) The target column "Perform" gives the value of risk-return performance for a period after the announcement. Those values ranges between -0.5 and 0.5. Small negative values correspond to the "sell" class, large positive values correspond to the "buy" class, and values close to 0 correspond to the "hold" class.

IV. Data Preprocessing

Before starting the training, we needed to address a few existing issues with the datasets.

A. Missing Values

There are two distinct types of missing values in the provided datasets:

1) Non-available: To fill these empty spaces with values that minimize the difference from actual unknown data, we opted to use the mean of the column values.
2) Non-applicable: We set these values to 0. Using any other value might suggest to our neural network model that these components should be active.

These two types were marked differently. Non-available data were represented as empty strings, while non-applicable data were marked with the "NA" string. This distinction allowed us to easily differentiate between the two cases.

B. Categorical Values

As mentioned earlier, the first column is a categorical variable indicating the company's sector, with eleven possible values. To correctly process this data, we one-hot encoded this column. This increased the number of columns in our datasets by ten, resulting in a total of 129 columns.

C. Standardization

To improve the performance of our model, we decided to normalize the datasets using z-score normalization [21], which calculates the standard score for each feature. The standard score, or z-score, of a feature x is calculated as:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of the feature values in the training set, and σ is the standard deviation of the feature values in the training set.

By applying z-score normalization, each feature will have a mean of 0 and a standard deviation of 1 in the training set. This transformation ensures that all features are on a similar scale, which can help improve the convergence speed of optimization algorithms and prevent features with larger scales from dominating the learning process.

The scaling parameters (mean and standard deviation) computed on the training set are then used to transform the test set, ensuring consistency in scaling across both datasets.

V. Model Architecture

We will test three neural networks with varying depths to see how well each of them manages the prediction task. All models will share common hyperparameters:

- Activation function: Hyperbolic Tangent (Tanh)
- Number of epochs: 100
- Learning rate: 0.00005
- Batch size: 40
- Optimizer: AdamW
- Loss function: mean squared error (MSE)

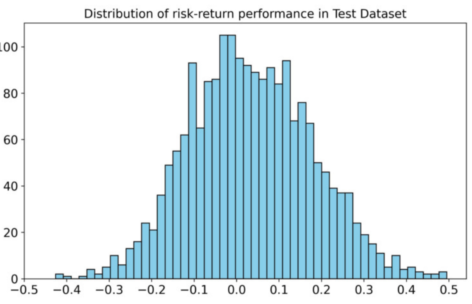

Fig. 1. Histogram of values from column "Perform" from Test Dataset

- Dropout probability: 20%
- Hidden layers size: 256

Each model accepts an input vector of 127 dimensions, which is processed through multiple dense hidden layers with dropout for regularization. To improve performance, we employed Xavier Glorot initialization [22] for our models weights. It works by initializing the weights using the formula:

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right)$$

where n_{in} and n_{out} are the size of input and output units in the layer.

Batch normalization [23] is applied after each layer to stabilize training. As mentioned earlier, the models vary in terms of the number of hidden layers. Specifically, the first model consists of 2 hidden layers, the second model includes 6 hidden layers and the third model incorporates 10 hidden layers. These configurations were chosen to explore how increasing depth impacts the model's ability to learn and generalize from the data.

The final layer of each model predicts a single value corresponding to the risk-return performance (target column "Perform"). Given that these performance values typically range between -0.5 and 0.5, we opted for the Hyperbolic Tangent activation function because it confines outputs to the range [-1, 1]. For a clearer visualization of these values, Fig 1 illustrates their distribution.

Our models are well-suited for the regression task of predicting risk-return performance. To adapt them for a classification problem, where each input should be assigned to one of the three classes described in previous sections, we implement the following post-processing steps:

We identify the highest value of risk-return performance from train dataset that still belongs to the class -1, marking this value as low_limit. Similarly, we identify the smallest value of risk-return performance from train dataset that corresponds to the class 1, marking this value as $upper\_limit$. The classification proceeds as follows:

- if model predicts value that is less than low_limit, then we classify this instance as class -1
- if it predicts value from range $[low\_limit, upper\_limit]$, then we classify it as class 0
- in case we get a value bigger than $upper\_limit$, then we classify it as class 1

The values of these boundaries rounded to five decimal places are:

$$low\_limit = -0.01504$$
$$upper\_limit = 0.04008$$

VI. Experimental Results

In this section, we present the results of our experiments. The models are listed in the order they were introduced in the previous sections (Model 1 has 2 hidden layers, Model 2 has 6 hidden layers and Model 3 has 10 hidden layers).

A. Evaluation metrics

Our model can be assessed using two metrics. The first metric is the mean absolute error (MAE) [24] between the predicted values of our model and the true values of risk-return performance. We opt for MAE over mean squared error (MSE) due to the small magnitude of values involved in our predictions. While MSE could provide useful insights as well, we prioritize MAE for its straightforward interpretation, especially when presenting results with fewer decimal points. The MAE is calculated as the average of the absolute differences between the true values y_i and the predicted values $\hat{y}_i$:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

The second metric, which we will call classification weighted error, corresponds to the evaluation criterion used in the FedCSIS 2024 Data Mining Challenge. After predicting classes, a confusion matrix is constructed. The evaluation involves calculating the average error cost using a predefined error cost matrix [2]:

	-1	0	1
-1	0	1	2
0	1	0	1
1	2	1	0

The classification weighted error is determined using the following formula:

$$\frac{1}{\text{length}}\sum(\text{cost matrix} \circ \text{confusion matrix})$$

Here, $\circ$ denotes the Hadamard product (element-wise multiplication), the term "length" represents the total number of predictions (2000 in our case) and the sum is over all the elements of the resulting matrix.

This method imposes a higher penalty for misclassifications between classes 1 and -1, specifically when predicting 'invest' for a true class of 'do not invest', and vice versa.

B. Achieved Scores

The table I displays the scores achieved for the two aforementioned metrics.

[2]https://knowledgepit.ai/fedcsis-2024-challenge/

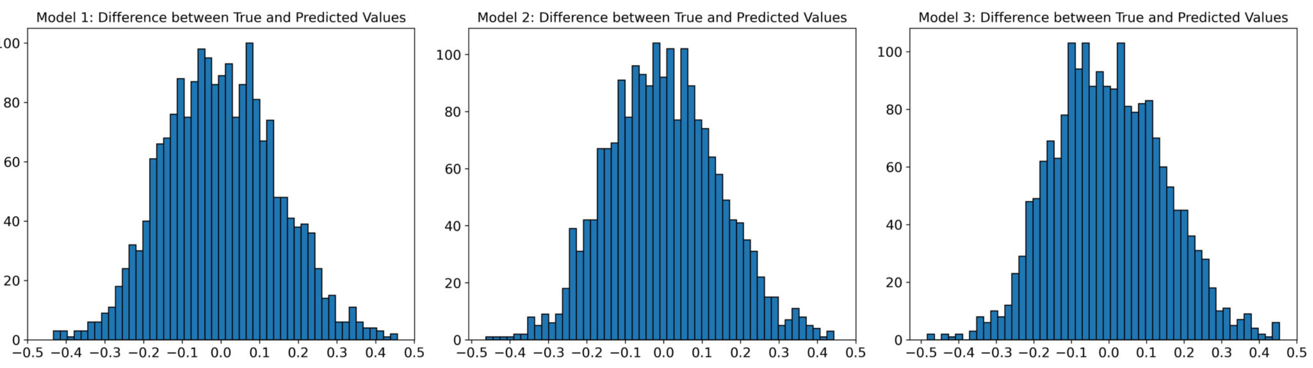

Fig. 2. Histograms of differences between true values of risk-return performance and predicted values

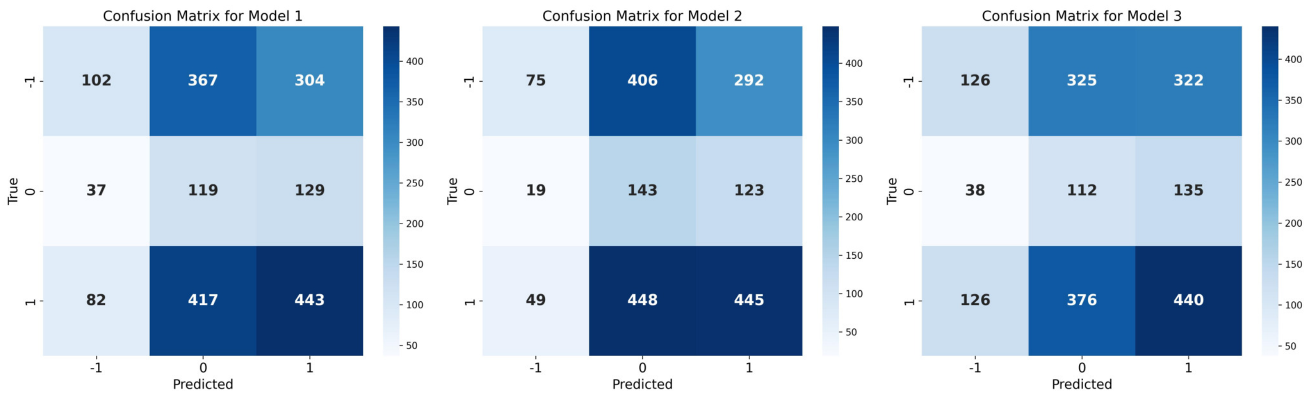

Fig. 3. Confusion matrices for all three models

TABLE I
ACHIEVED SCORES ROUNDED TO FOURTH DECIMAL POINT

Model	MAE	Classification Weighted Error
Model 1	0.1174	0.861
Model 2	0.1158	0.839
Model 3	0.1197	0.885

Fig 2 presents histograms of differences between true values and predicted values. Although the distributions appear quite similar, these slight variances lead to marginally different outcomes in those two aforementioned metrics.

In Fig 3, the confusion matrices for each model are presented. We observe consistent patterns across all three models, such as managing to correctly classify a large number of ’1’s while also frequently misclassifying classes -1 and 1 as 0. Although adjusting the values of previously mentioned low_limit and $upper\_limit$ might seem like a logical step to reduce the frequency of classifying instances as 0, this adjustment, most of the times, does not significantly improve the weighted classification error. This is because the error metric penalizes misclassifications between classes -1 and 1 more severely. Therefore, by allowing the model to maintain such a buffer of class "0", we are empirically able to achieve better results in terms of weighted classification error.

Although our models achieved comparable results, it is evident that hyperparameter selection is a crucial aspect of our method. Analyzing the outcomes, we observed that the smallest model slightly underfitted while the largest one overfitted. Therefore, despite each model achieving a fairly good level of performance, fine-tuning them requires a thorough grid search.

VII. CONCLUSIONS

In this study, we proposed a robust framework for predicting stock market trends using feedforward neural networks. We aimed to enhance the accuracy of forecasting models by leveraging advancements in deep learning and extensive preprocessing techniques. Our approach involved tackling challenges such as handling large datasets of financial statements, managing missing data, and preprocessing categorical variables and numerical features. We explored three neural network architectures with varying numbers of hidden layers, evaluating their performance based on both regression (mean absolute error) and classification (weighted error) metrics. This approach achieved best cumulative performance in terms of the risk-return aspect in FedCSIS 2024 Data Mining Challenge.

REFERENCES

[1] P. Tran, P. Anh, P. Tam, and C. Nguyen, “Applying machine learning algorithms to predict the stock price trend in the stock market – the case

of vietnam," *Humanities and Social Sciences Communications*, vol. 11, 03 2024.

[2] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[3] N. Burton, *An Analysis of Burton G. Malkiel's A Random Walk Down Wall Street*. Macat Library, 01 2018.

[4] M. Agrawal, A. Khan, and P. Shukla, "Stock price prediction using technical indicators: A predictive model using optimal deep learning," *International Journal of Recent Technology and Engineering*, vol. 8, pp. 2297–2305, 07 2019.

[5] W. R. Kinney, "Arima and regression in analytical review: An empirical test," *The Accounting Review*, vol. 53, no. 1, pp. 48–60, 1978.

[6] G. Sonkavde, D. S. Dharrao, A. M. Bongale, S. T. Deokate, D. Doreswamy, and S. K. Bhat, "Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications," *International Journal of Financial Studies*, vol. 11, no. 3, 2023.

[7] A. M. Rakicevic, P. D. Milosevic, I. T. Dragovic, A. M. Poledica, M. M. Zukanovic, A. Janusz, and D. Slezak, "Predicting stock trends using common financial indicators: A summary of fedcsis 2024 data science challenge held on knowledgepit.ai platform," in *Proceedings of FedCSIS 2024*, 2024.

[8] J. Zheng, D. Xin, Q. Cheng, M. Tian, and L. Yang, "The random forest model for analyzing and forecasting the us stock market in the context of smart finance," 2024.

[9] T. Strader, J. Rozycki, T. Root, and Y.-H. Huang, "Machine learning stock market prediction studies: Review and research directions," *Journal of International Technology and Information Management*, vol. 28, pp. 63–83, 01 2020.

[10] Z. Liu, Z. Dang, and J. Yu, "Stock price prediction model based on rbf-svm algorithm," in *2020 International Conference on Computer Engineering and Intelligent Control (ICCEIC)*, pp. 124–127, 2020.

[11] S. Mehtab, J. Sen, and A. Dutta, "Stock price prediction using machine learning and lstm-based deep learning models," in *Machine Learning and Metaheuristics Algorithms, and Applications* (S. M. Thampi, S. Piramuthu, K.-C. Li, S. Berretti, M. Wozniak, and D. Singh, eds.), (Singapore), pp. 88–106, Springer Singapore, 2021.

[12] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[13] J. Eapen, D. Bein, and A. Verma, "Novel deep learning model with cnn and bi-directional lstm for improved stock market index prediction," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0264–0270, 2019.

[14] H. Kaeley, Y. Qiao, and N. Bagherzadeh, "Support for stock trend prediction using transformers and sentiment analysis," 2023.

[15] R. Kumar, C. M. Sharma, V. M. Chariar, S. Hooda, and R. Beri, "Emotion analysis of news and social media text for stock price prediction using svm-lstm-gru composite model," in *2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, pp. 329–333, 2022.

[16] E. Kannout, M. Grzegorowski, and H. Son Nguyen, *Toward Recommender Systems Scalability and Efficacy*, pp. 91–121. Cham: Springer International Publishing, 2023.

[17] E. Kannout, M. Grzegorowski, M. Grodzki, and H. S. Nguyen, "Clustering-based frequent pattern mining framework for solving cold-start problem in recommender systems," *IEEE Access*, vol. 12, pp. 13678–13698, 2024.

[18] Sharma, Vikram, Rakhra, Manik, and Mathur, Gauri, "Hybrid approaches for stocks prediction and recommendation system," *E3S Web Conf.*, vol. 453, p. 01047, 2023.

[19] E. Kannout, H. S. Nguyen, and M. Grzegorowski, "Speeding up recommender systems using association rules," in *Intelligent Information and Database Systems* (N. T. Nguyen, T. K. Tran, U. Tukayev, T.-P. Hong, B. Trawiński, and E. Szczerbicki, eds.), (Cham), pp. 167–179, Springer Nature Switzerland, 2022.

[20] P. Paranjape-Voditel and U. Deshpande, "An association rule mining based stock market recommender system," in *2011 Second International Conference on Emerging Applications of Information Technology*, pp. 21–24, 2011.

[21] N. Fei, Y. Gao, Z. Lu, and T. Xiang, "Z-score normalization, hubness, and few-shot learning," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 142–151, 2021.

[22] Y. Bengio and X. Glorot, "Understanding the difficulty of training deep feed forward neural networks," *International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 01 2010.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.