Predicting Financial Time Series for Value Investment

Krasin Georgiev[1], Kiril Koparanov[2], Daniela Minkovska[3*]

[1,3]*Associate Professor, PhD, Technical University of Sofia, Department of Aeronautics, Sofia, Bulgaria; E-mail: daniela@tu-sofia.bg*

[2]*Assistent Professor, PhD, Technical University of Sofia, Department of Programming and Computer Technologies, Sofia, Bulgaria*

Abstracts: Value investment is an attractive paradigm for individual investors. It involves different steps including evaluating past performance that could be challenging. We propose a representation for financial time series in a form appropriate for both human interpretation and automatic processing. We design a model for predicting sequence of values as opposed to point values. Combined with application of encoder-decoder type of neural network model architecture this allows interpretation of model parameters and intermediate activations by domain experts. We show that predictions better than the trivial last observed value are possible. Therefore, informed investment decisions can be supported by neural network models and the proposed representation and model interpretation.

Keywords: Time Series, Sequence Prediction, Stock Prices, Neural Networks, Value Investment, Forecasting.

1. INTRODUCTION

Time sequence prediction is a broad field of research with applications in many different areas [1]. Some examples include patient monitoring (Medicine and healthcare); renewable energy forecasting, climate modelling, and pollution prediction (Energy and environmental science); traffic prediction, demand forecasting (Transportation and logistics); sales forecasting, market trend analysis (Marketing and sales); quality control, inventory management (Manufacturing and supply chain management). As the amount of data generated by various industries and applications continues to increase, the need for accurate time sequence prediction models will also grow.

Stock market prediction and financial time series analysis are important applications of time sequence prediction. Many studies have been conducted in this area, with the aim of accurately forecasting stock prices, identifying trends, and detecting anomalies in financial time series data [2, 3].

There are two contradictory and complementary assumptions done when the market is modelled - about stationarity and about growth. Assuming stationary time series means that there is some intrinsic (unknown) value of the stock and random fluctuations around it. The risk is related to the possibility to buy high and sell low. As average return is zero, the risk premium is related to the price of the uncertainty and is calculated as the variance (or standard deviation) of the stock prices [4]. Assuming non-stationary time series means that there is a second independent property of the stock, the growth rate or the expected return per period. The growth investment relies on this “growth” of the mean value of the stock. Unfortunately, it is unknown and also related to the investment risk.

Value investing and growth investing differ in their approach to risk, with value investing focusing on finding undervalued stocks and growth investing seeking high-growth potential companies. While both strategies have their own risks, they both require some kind of market predictions, trying to catch the systematic and unsystematic factors as well as the intrinsic value, volatility and growth. It is likely that the input and target data preprocessing steps as scaling, filtering, trend removal, input features, etc. will be different though.

Statistical models allow generation of forecasts as a function of time, accounting for different factors, online evaluation of the forecast performance and manual adjustment [5]. Machine learning (ML) approaches give good results for single point predictions of a binary class, e.g. growth/decline, above/below, etc. Simple models as regression, multi-layer perceptron and Hidden Markov Models can beat the market and provide a gain greater than one [6]. A type of network that is widely explored is the Long Short Term Memory [7, 8].

Most published results are based on small datasets created for one or several stocks [9, 10]. Main emphasize is on the models, prediction algorithm and evaluation metrics [3]. The strength though of the ML is the ability to train a single model on many series from a big and diverse datasets (cross-learning, [11]). This imposes special requirements to the input data preprocessing step to make learning possible [11]. It could be integrated in the model, e.g. exponential smoothing, combined with LSTM in a hierarchical main model [11, 12].

Previous attempts of the authors to model financial time series, while showing some insights, proved to be unsuccessful with poor predictions that were worse than predicting last observed value [13]. The same is true for other published and unpublished studies. At the same time the widely spread practice in technical analysis to look for specific data patterns assume that such predictive patterns exist. In general, we don't know if persistent patterns in data exist to be modelled. If a model exists, we don't know how to find adequate (if not best) parameters of the model. Training a neural network model is not trivial – the nature of the existing approaches require specific format of the inputs and outputs.

We will explore the capabilities of modestly complex neural networks to predict a sequence of future values. Emphasize will be on input and output series representation that allow mixing data from multiple companies. These can be applied directly to enhance the other approaches developed and studied in the literature. Moreover it is already an established practice to first design and train a complex and universal models and then to fine tune on small datasets. This so called transfer learning can be done for an individual stock or groups of stocks.

2. NEURAL ARCHITECTURES FOR TIME SERIES FORECASTING

We explore a class of architectures that predict a time series as a sum of weighted base sequences. The model learns the base sequences and how to calculate the weights needed to combine them. The network has two parts in a sequence - head and body:

$$y = \mathrm{NN}(x) = \mathrm{NN}_{\mathrm{head}}[\mathrm{NN}_{\mathrm{body}}(x)] \qquad (1)$$

where "NN" denotes the whole model, "$\mathrm{NN}_{\mathrm{body}}$" is the body part and "$\mathrm{NN}_{\mathrm{head}}$" is the head part, *x* is the input sequence and *y* is the output sequence (see Fig. 1 as an example).

$$x_n = [s_{n-L-1}, \dots, s_n], x_n \in \mathbb{R}^{L \times F}$$
$$y_n = [s_{n+1}, \dots, s_{n+K}], y_n \in \mathbb{R}^{K}$$

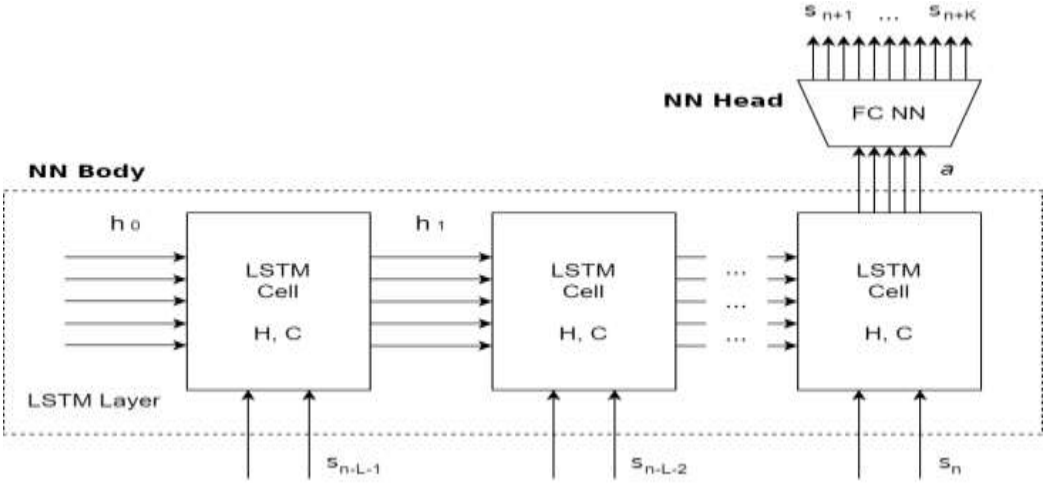


Figure 1. Neural architecture for time series forecasting with body and head parts. Example with LSTM body with L cells and H output features. The head is fully connected layer with K outputs.

The weights are the output activations *a* of the body part of the model. "NN_{body}"' can be any neural network that process a sequence of vectors into a sequence of numbers, e.g.:

- simple linear transformation, after flattening (LIN)
- fully connected feed forward network, after flattening (FC)
- multi-layer convolutional neural network followed by average pooling (CNN)
- multi-layer convolutional neural network followed by flattening (CNNb)
- Long Short-Term Memory networks (LSTM) - Fig. 1
- advanced architectures, e.g. one dimensional adaptation of ResNet (RES), convolutional layers followed by LSTM layer (CLSTM), etc.

$$a = \mathrm{NN}_{\mathrm{body}}(x) = \mathrm{LSTM}(x)$$

The base sequences are the parameters of the head part of the model. The head of the model is an ordinary linear layer (fully connected, single layer, no nonlinear activation network):

$$y = \mathrm{NN}_{\mathrm{head}}(a) = a \cdot A + b$$

where *a* provides the weights of the sequences (row vector) and **A** is the transformation matrix (each row is a base sequence that will be scaled by the corresponding weight activation) and *b* is the bias vector, a base sequence that is independent of the input data.

3. MATERIALS AND METHODS

Data preparation is important part of any empirical research that determines its chances for success to a greet degree. We start with a short description of the data source and data pipeline. More details on the transformation of the training and target sequences follow. This is the main contribution of the paper that is subject of further study. The procedure for model evaluation is also provided.

3.1. Data files, Data sets and Data loaders

Data files for 7276 Stock symbols (ticker symbols) were downloaded from Yahoo Finance on 5 Jan 2023 via yfinance python package (v0.2.3, https://pypi.org/project/yfinance/). Each file contains daily quotations with seven fields - Date, Open, High, Low, Close, Adj Close and Volume. All files were loaded in single data structure - pandas (v1.5.2) dataframe where an additional field with Ticker symbol was added. The complete dataframe was split into three subsets based on date - train (years 1962-2015, 15445343 records), validation (years 2016, 2017 and 2018, 3188695 records) and test (years 2019-2022, 5852575 records).

A Dataset capable to extract historic and target sequences of predefined lengths (look back and prediction horizon lengths) was designed (process called windowing and mapping data) and loaded with train, validation and test data. The index of the start record of a sequence was selected in a way not to mix data from different stocks. Incomplete sequences were discarded. Therefore the number of trains, validation and test sequences varies for different lengths of the look back and prediction horizon. Finally, data were grouped into batches of 2048 using Dataloaders to speed up calculations. The same training batch was used three times before generating new one for both speed and performance improvements [14]. The training Dataloader also provided shuffling of the data.

3.2. Return and Log-Return or what to model?

Predicting the stock prices is usually done in the context of investing for profit. Therefore, the final goal is predicting the return from an investment for a known or unknown period of time.

Returns *R* are defined based on the stock prices *P*:

$$R_{i+1} = \frac{P_{i+1} - P_i}{P_i} \tag{2}$$

For small values of the returns they can be conveniently replaced by log-returns:

$$r_{i+1} = \ln(R_{i+1} + 1) \approx R_{i+1} \tag{3}$$

Logarithms have advantages for viewing economic data. First they allow a wider dynamic range to be visualized so the volatility at lower values is not hidden. Second, they are also directly related to the returns, i.e. the change in log-price is informative on its own while the change in price have value only if compared to the initial price. Using log-returns with regard to simple returns is more convenient for mathematical manipulations – operations are symmetrical, and log-returns are additive.

$$r_{i+1} = \ln \frac{P_{i+1}}{P_i} = \ln P_{i+1} - \ln P_i$$

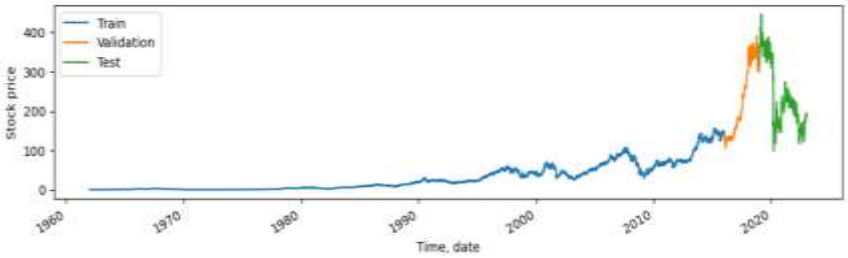
It is hard to learn patterns from the raw stock prices because different sequences are not on the same scale. They can range from values close to zero to thousands not only for different stocks but even for the same stock in different time periods (Fig. 2a). Transforming raw stock prices by taking logarithm improves the picture (Fig. 2b)).

Alternative representations of two stock price sequences are shown in Fig. 3. Value difference-based transformation, e.g. per step price difference, normal returns (2) and log-returns (3), look more stationary and vary in a narrow range, which is a prerequisite for modeling. On the other hand, these does not reveal the trend and are hard to interpret by human (Fig. 3c).

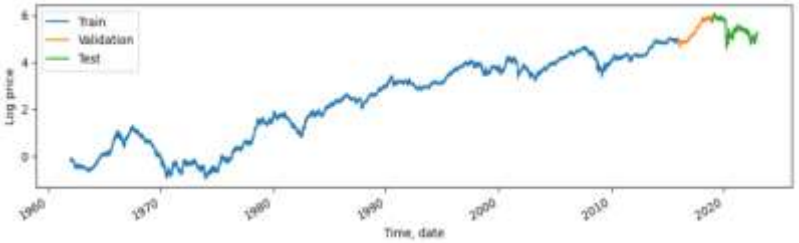
Our final solution shifts the log-sequence so the last observed value is zero and thus adds the following features (Fig. 3d):

$$r_{i(0)} = \ln P_i - \ln P_0 = \ln \frac{P_i}{P_0} \tag{4}$$

- targets/predictions are log-returns with regard to the last observed value and can be interpreted directly
- sequences are centered around zero which is advantage for model training
- training history is unique and different for each time step which is advantage for preventing overfitting during model training
- shifting the sequence up and down could be used for data augmentation as the pattern is preserved (only the zero for the return is changed)

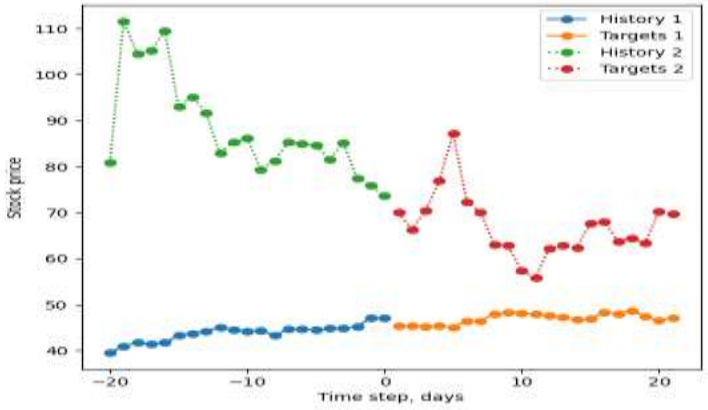


(a)

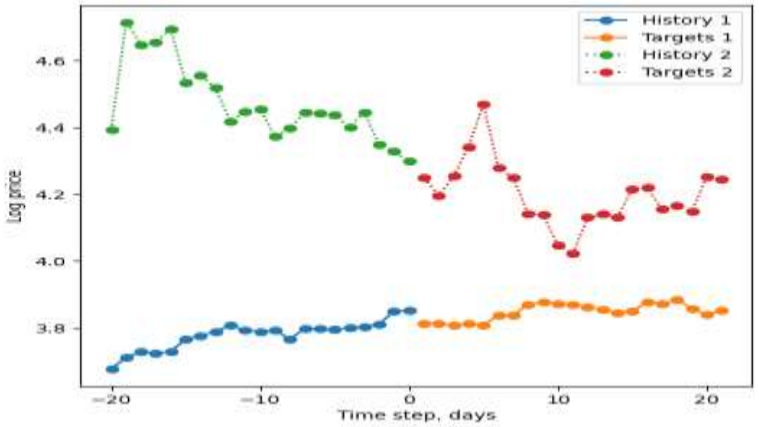


(b)

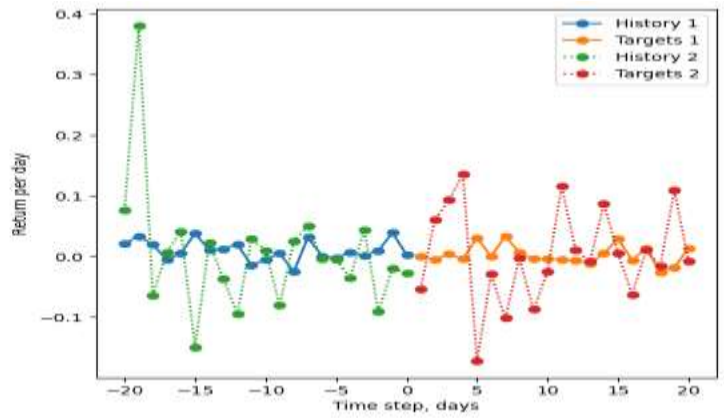
Figure 2. Stock prices for Boeing company (**a**) Raw stock price in USD. (**b**) Log stock price.



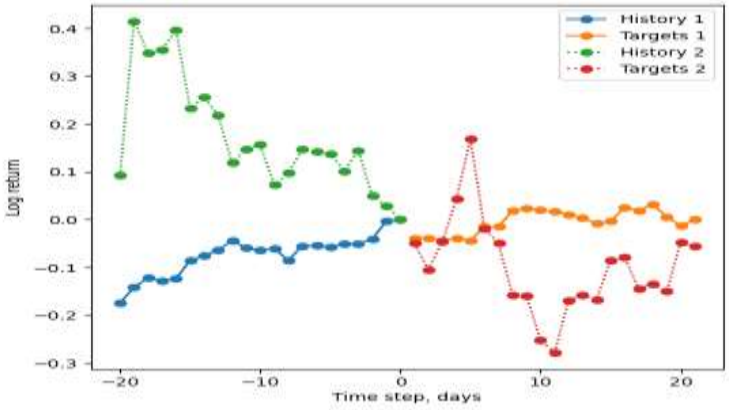
(a)



(b)



(c)



(d)

Figure 3. Stock price representations (**a**) Raw stock price in USD. (**b**) Log stock price. (**c**) Daily returns. (**d**) Log-return relative to the last observed value. The negative values on the x-axis represent the look back sequence and the positive values - the target sequence.

3.3. Training the Model

All models were implemented and trained using Pytorch (v. 1.13.1) and Fastai (v. 2.7.10) frameworks. Weight initialization, optimizer, learning rate schedule, etc. were left to their default values. Training was done with Fastai one cycle training routine and the corresponding default learning rate scheduler [15, 16].

The mean squared error (MSE) was chosen as the loss function, which is a common practice in regression problems:

$$L = \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\text{target}_{nk} - \text{prediction}_{nk}\right)^2 = \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K} L_{nk} = \frac{1}{N}\sum_{n=1}^{N} L_n \quad (5)$$

where n is the index of the sample in the dataset, L_n is the loss for the n-th sequence and k is the index in the sequence of the target. The loss for each step in the prediction is denoted as L_{nk}.

After a small number of initial tests, all training hyperparameters were preselected and then fixed and not tuned further:

- learning rate was 0.01;
- the number of training epochs was 4 x 3 = 12 (four passes through the dataset but each mini-batch is processed three times before going to the next one [14]);
- model architecture (number of nodes, layers and their arrangement) – according to the considerations provided in Sec. 2

3.4. Evaluating the Results

The MSE of the shifted log-returns was chosen as an evaluation metric. When the targets are log-prices ($\ln P_k$) and shifted log-returns ($r_{k(0)} = \ln P_k/P_0$) their losses are comparable, and the loss (5) can be used as a metric to measure model performance with

$$L_{nk} = \left(r_{nk(0)} - \hat{r}_{nk(0)}\right)^2 = \left(\ln\frac{P_{nk}}{P_{n0}} - \ln\frac{\hat{P}_{nk}}{P_{n0}}\right)^2 = \left(\ln P_{nk} - \ln\hat{P}_{nk}\right)^2$$

When the targets are raw prices (P_k) or differences based (e.g. $\ln P_{k+1} - \ln P_k$), additional transformations are needed to obtain a metric that is comparable to the above one, logarithm and integration, respectively. Scaling of the raw prices to a preselected range does not change the metric and the results are still comparable.

The variance and the standard error of the evaluation metric can be used as a measure of its uncertainty:

$$\mathrm{var}(L_n) = \frac{1}{N}\sum_{i=1}^{N}(L_n - L)^2$$

$$\mathrm{se}(L_n) = \sqrt{\frac{\mathrm{var}(L_n)}{N}}$$

Trained model performance metrics on the validation and the test datasets were compared to the metrics for the baseline model on the corresponding datasets.

$$d_n = L_{n\ \mathrm{Baseline}} - L_{n\ \mathrm{Model}}$$

$$\mathrm{mean}(d_n) = L_{\mathrm{Baseline}} - L_{\mathrm{Model}} \to d$$

$$\mathrm{var}(d_n) = \frac{1}{N}\sum_{n=1}^{N}(d_n - d)^2$$

$$\mathrm{se}(d_n) = \sqrt{\mathrm{var}(d_n)/N}$$

Model performance was also expressed relative to the baseline model performance:

$$\bar{L}_{\mathrm{Model}} = \frac{L_{\mathrm{Baseline}} - L_{\mathrm{Model}}}{L_{\mathrm{Baseline}}} \cdot 100$$

This MSE reduction representation of the error is useful for performance comparison and visualization, e.g. for different models, different preprocessing, etc. Positive value means that the result is better than the baseline and the error is reduced. Higher value is better than lower.

There are no well-established methods for evaluation of significance of the differences of neural network predictions for sequences. The statistical significance of the difference between the baseline and the models was first evaluated assuming independence of the samples and normal distribution of the performance metric through dependent t-test for paired samples. The T-statistic is the ratio of the mean and the standard error of the difference between all pairs of baseline and model losses:

$$\mathrm{T\_stat} = \frac{\mathrm{mean}(d_n)}{\mathrm{se}(d_n)}$$

The test recommended for sequential data is the Diebold-Mariano Test (DM test) [17, 18]. It uses different estimate of the standard error which accounts for the correlation between the predictions.

$$\mathrm{DM\_stat} = \frac{\mathrm{mean}(d_n)}{\mathrm{se}_{\mathrm{DM}}(d_n)}$$

$$\mathrm{se}_{\mathrm{DM}}(d_n) = \sqrt{\frac{1}{N^2}\sum_{\tau=-\infty}^{\infty}\sum_{n=|\tau|+1}^{N}(d_n - d)(d_{n-|\tau|} - d)}$$

where τ is the lag and the sum is truncated to $\tau = \sqrt[3]{N} + 1$.

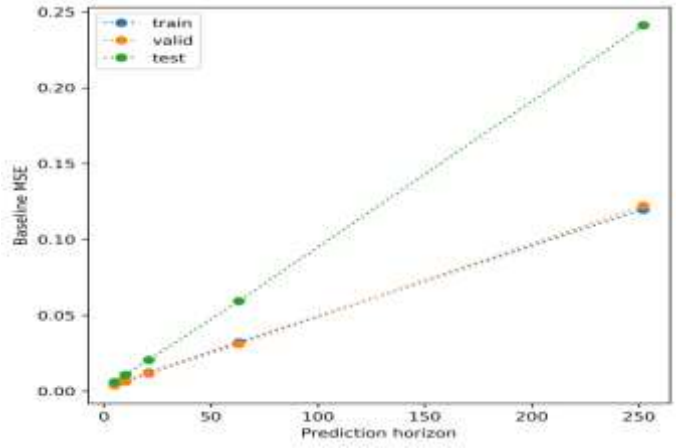
The learning ability of each model was evaluated based on the performance on the train dataset. Predictions on the validation dataset were used to check for overfitting. They were not used for hyper-parameter tuning and therefore together with the test dataset predictions provide a metric for the selected prediction error. The performance on the test dataset gives information about the robustness of the model for a different data distribution.

4. RESULTS

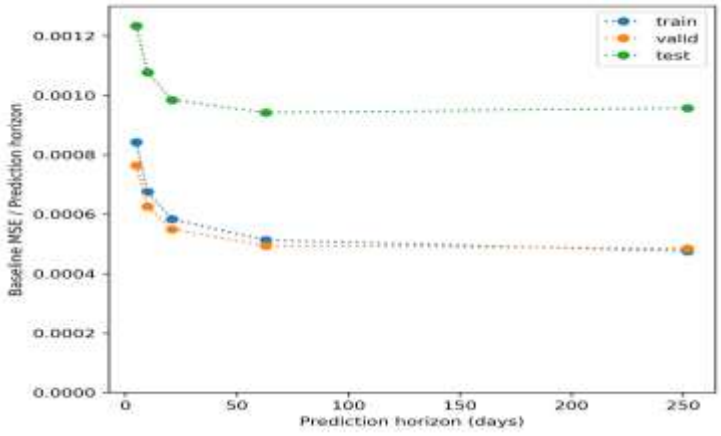
We evaluate the ability of predicting sequences as a whole in the context of forecasting stock market prices. Models based on the architectures described in Sec. 2 “Neural Architectures for Time Series Forecasting” were trained using the training dataset and different pre-processing as described in Sec. 3 “Materials and Methods” (Subsec. 3.1 and 3.2 respectively). The models based on standard architectures with 32 hidden features will be further called “core” models (LIN, FC, CNN, CNNb and LSTM). Some other “advanced” models were also compiled (RES, RESb, CLSTM).

4.1. Baseline model

The baseline is predicting the last observed value. This is equivalent to zero return for all future steps. The MSE of this prediction is related to the variance of the target values ($\mathrm{Var}[X] = \mathrm{E}[X^2] - (\mathrm{E}[x])^2$, $\mathrm{MSE}[X] = \mathrm{Var}[X] + (\mathrm{E}[x])^2$). The prediction error as a function of the prediction horizon is shown in Fig. 4. As can be expected it increases with the increase of the prediction horizon. The relationship is close to linear as the variance of a sum of independent variables is the sum of the variances of all variables.



(*a*)



(*b*)

Figure 4. Performance (MSE) of the baseline model (**a**) Linear scale. (**b**) Scaled by the number of prediction steps ahead (prediction horizon).

4.2. Predictions for different input representations

The model predictions for different representations of the data sequence (different preprocessing) were evaluated for the core models on both the validation and the test datasets. The results are summarized in Table 1 where the MSE of the LSTM model is shown together with an estimate of its standard error, the difference with regard to the baseline model (column Delta) is shown together with statistics that describe the significance of the difference (both T-test and DM-test).

The raw data (preprocessed with simple scaling) does not allow the model to be trained properly – the performance is worse than the baseline. Using per step returns data improves training but the difference from the baseline is still not significant. Using simple logarithm gives improved performance on the validation dataset that is significant but it is not achieved on the test dataset. The proposed representation provides improved performance relative to the baseline and the difference is both significant and robust to changes in data distribution – the performance is improved on both validation and test datasets.

More details about the performance of the different models are shown in Fig. 5. This visualization allows model emerging properties as trainability, overfitting and robustness to be easily compared. These were evaluated for the studied input representations and again the proposed representation provided the best properties.

Table 1. LSTM model performance. Delta is the improvement of MSE relative to the baseline. SE is for standard error The baseline MSEs are 1224.6e-5 (train), 1153.4e-5 (validation) and 2067.0e-5 (test)

DS	Format	MSE, *L*		Delta, *d*			T test		DM test		Note
		mean	SE	mean	SE	SE DM	T stat	p-val	DM stat	p-val	
Val	Raw price	5643e-5	97.0e-5	-4.5e-2	96.8e-5	1.4e-2	-46.4	1.0	-3.2	1.0	Failed
	Per step return	1153e-5	7.4e-5	0.5e-5	1.7e-5	12.6e-5	0.29	0.38	0.04	0.48	Same
	Log price	1134e-5	6.9e-5	19.6e-5	1.9e-5	2.9e-5	10.5	6e-26	6.8	6e-12	Improved
	Log-return	1139e-5	7.3e-5	14.1e-5	1.17e-5	4e-5	12.0	1e-33	3.5	2e-04	Improved
Test	Raw price	4135e-5	13.1e5	-2068e-5	12.0e-5	113e-5	-173	1.0	-18	1.0	Failed
	Per step return	2063e-5	6.4e-5	3.8e-5	1.3e-5	4.2e-5	2.8	2e-3	0.9	0.18	Same
	Log price	2061e-5	6.6e-5	6.1e-5	3.7e-5	11.2e-5	1.6	5e-2	0.54	0.29	Same
	Log-return	2047e-5	6.4e-5	20.1e-5	1.4e-5	3.4e-5	14.7	3e-50	5.9	1.9e-9	Improved

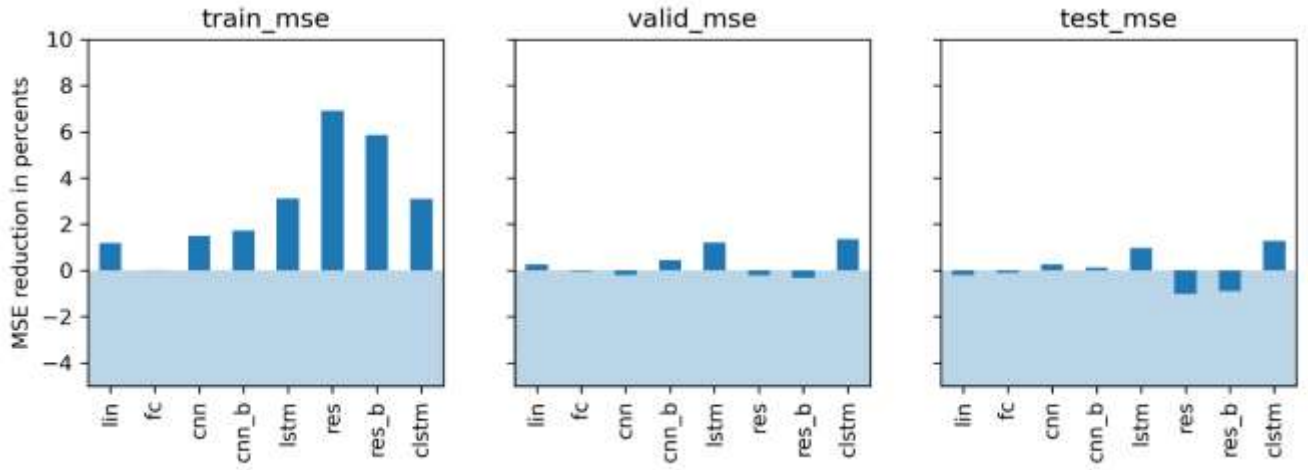


Figure 5. MSE reduction ($\bar{L}_{Model}$) for generic predictions of core and advanced models, prediction horizon of 21 steps. Train (left), validation (center) and test (right) dataset.

4.3. Predictions for different prediction horizons

The predictions are consistently better than the trivial ones when evaluated for different prediction horizons. Figure 6 shows that for short sequences, e.g. 5, 10, 21, most core models provide improved performance on both validation and test dataset. With longer sequences there is at least one model better than trivial.

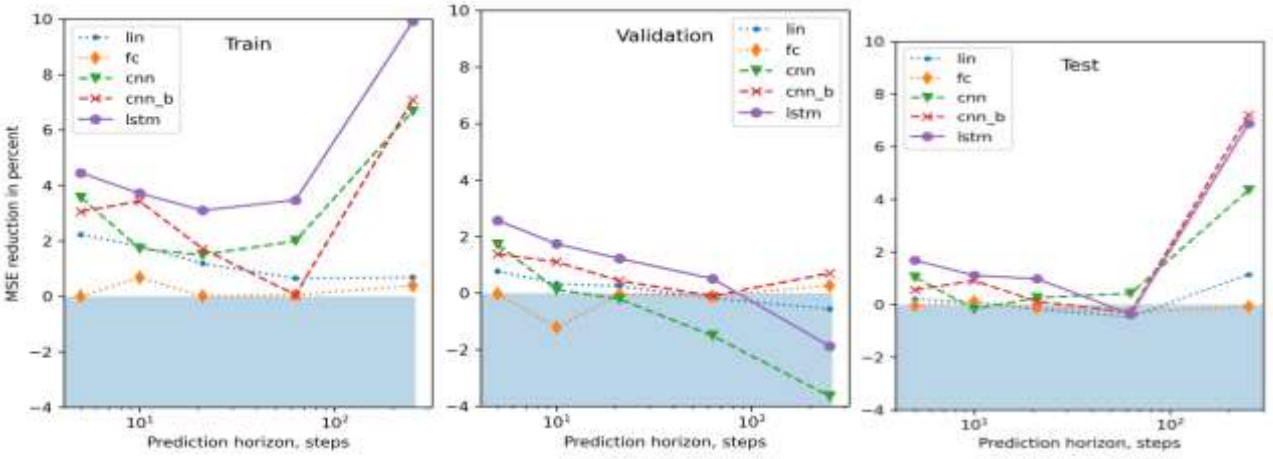


Figure 6. Core models performance improvement for different prediction horizons. Train (left), validation (center) and test (right) dataset

CONCLUSION

It was shown that the proposed log-return representation in addition to the explicit output interpretation provides better training than the alternatives discussed. It is robust to outliers and mixed stocks the prices of which differ in orders of magnitudes. This makes possible training with huge datasets. Therefore the models created in this way are universal and can be used to discover general patterns.

The achieved performance while not impressive is better than the trivial baseline and the difference was shown to be statistically significant. The train, validation and test datasets cover very different time ranges and economical periods. Therefore, obtaining a positive result is promising for the potential of the developed approaches.

Improvements of time sequence predictions could be achieved through the use of bigger and customized machine learning models implementing state of the art in the field; incorporation of more diverse and relevant data sources, such as economic indicators, fundamental company data, news and social media activity, etc.; development of more robust loss functions and evaluation metrics that take into account the specific characteristics of time series data; replacing predictions with sequence generation, e.g. the ChatGPT of OpenAI is a striking example of what is possible in generation of meaningful text and refinement for a specific context. Important area of development is to model the uncertainty of the prediction of the model. Ideally this will add an output related to the reliability of the prediction and highlight cases when a special pattern in the time series is observed.

ACKNOWLEDGEMENTS

The authors would like to thank the Research and Development Sector at the Technical University of Sofia for the financial support.

REFERENCES

[1] Salinas, D., Flunkert, V., Gasthaus, J. & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting, 36(3), 1181-1191.

[2] Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications : A survey. Applied Soft Computing, 93, 106384. https://doi.org/https://doi.org/10.1016/j.asoc.2020.106384.

[3] Rouf, N., Malik, M. B., Arif, T., Sharma, S., Singh, S., Aich, S., & Kim, H. C. (2021). Stock Market Prediction Using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions. Electronics 2021, Vol. 10, Page 2717, 10(21), 2717. https://doi.org/10.3390/ELECTRONICS10212717

[4] Zhang, X.-P. S., & Wang, F. (2017). Signal Processing for Finance, Economics, and Marketing: Concepts, framework, and big data applications. IEEE Signal Processing Magazine, 34(3), 14–35. https://doi.org/10.1109/MSP.2017.2663138

[5] Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. The American Statistician, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[6] Rosas-Romero, R., & Medina-Ochoa, J.-P. (2019). Learning Financial Time Series for Prediction of the Stock Exchange Market. In G. Lee & Y. Jin (Eds.), Proceedings of 34th International Conference on Computers and Their Applications (Vol. 58, pp. 418–427). EasyChair. https://doi.org/10.29007/mh4m

[7] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654–669. https://doi.org/https://doi.org/10.1016/j.ejor.2017.11.054

[8] Singh, S., Ahmad, M., Bhattacharya, A., & Azhagiri, M. (2019). Predicting stock market trends using hybrid SVM model and LSTM with sentiment determination using natural language processing. International Journal of Engineering and Advanced Technology, 9(1), 2870–2875. https://doi.org/10.35940/ijeat.A1106.109119

[9] Agustini, W.F., Affianti, I.R. & Putri E. (2018). Stock price prediction using geometric Brownian motion. Journal of Physics: Conference Series, 974(1), 012047.

[10] Haq, I. U., Ramay, M. I., Rehman, M. A. U., & Jam, F. A. (2010). Big five personality and perceived customer relationship management. Research Journal of International Studies, 15, 37-45.

[11] Bal, C., Demir, S..(2017). Forecasting TRY/USD Exchange Rate with Various Artificial Neural Network Models. TEM Journal, 6(1), 11-16

[12] Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting, 36(1), 75–85. https://doi.org/10.1016/J.IJFORECAST.2019.03.017

[13] Smyl, S., Dudek, G., & Pelka, P. (2021). ES-dRNN: A Hybrid Exponential Smoothing and Dilated Recurrent Neural Network Model for Short-Term Load Forecasting. ArXiv Preprint ArXiv:2112.02663v1.

[14] Koparanov, K., Trifonov, R., Minkovska, D., & Georgiev, K. (2022). Forecasting a long sequence of values with a neural network using direct

and recurrent approaches. AIP Conference Proceedings, 2557, 050001. https://doi.org/10.1063/5.0103975

[15] Howard, J. (2022). A sneaky trick: keep GPU busy with MultDL. In Practical Deep Learning for Coders part 2: Deep Learning Foundations to Stable Diffusion, Lesson 20 [MOOC], fastai,, https://course.fast.ai/Lessons/part2.html

[16] Leslie N. Smith. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. arXiv:cs.LG/1803.09820

[17] Howard, J., & Gugger, S. (2020). Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD. O'Reilly Media, Incorporated.

[18] Diebold, F.X.; Mariano, R.S. (1995) Comparing Predictive Accuracy. Journal of Business & Economic Statistics, 13, 253–263. https://doi.org/10.2307/1392185.

[19] Diebold, F.X. (2015). Comparing Predictive Accuracy, Twenty Years Later: A Personal Perspective on the Use and Abuse of Diebold–Mariano Tests. Journal of Business & Economic Statistics, 33, 1–9. https://doi.org/10.1080/07350015.2014.983236.

DOI: https://doi.org/10.15379/ijmst.v10i3.1767