University of Warsaw
Faculty of Economic Sciences

Working Papers
No. 32/2020 (338)

FRACTIONAL DIFFERENTIATION AND ITS USE IN MACHINE LEARNING

Janusz Gajda
Rafał Walasek

Warsaw 2020

WORKING PAPERS 32/2020 (338)

Fractional differentiation and its use in machine learning

Janusz Gajda*, Rafał Walasek*

Faculty of Economic, University of Warsaw
** Corresponding authors: jgajda@wne.uw.edu.pl, rwalasek@wne.uw.edu.pl*

Abstract: This article covers the implementation of fractional (non-integer order) differentiation on four datasets based on stock prices of main international stock indexes: WIG 20, S&P 500, DAX, Nikkei 225. This concept has been proposed by Lopez de Prado to find the most appropriate balance between zero differentiation and fully differentiated time series. The aim is making time series stationary while keeping its memory and predictive power. This paper makes also the comparison between fractional and classical differentiation in terms of the effectiveness of artificial neural networks. This comparison is done in two viewpoints: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The conclusion of the study is that fractionally differentiated time series performed better in trained ANN.

Keywords: fractional differentiation, financial time series, stock exchange, artificial neural networks

JEL codes: C22, C32, G10

1. Introduction

Many supervised machine learning algorithms applied on financial time series data require stationarity. In the process of creating a predictive model it is assumed that a given time series is generated by a stochastic process. To make an accurate inference from such model it is crucial that mentioned data generation process to remain consistent. In statistical terms, the mean, variance and covariance should not change with time. As a result in analyzed time series a trend over time is not revealed. When this assumption is not fulfilled the algorithm may assign a wrong prediction to a new observation.

In contrary to cross-sectional data time series is a specific kind of data in the sense that any observation reflects a history of observations occurred in the past. The literature named it as a distinctive memory of past track record. Due to the stationarity condition this series memory is often excluded from the dataset.

The most commonly used method for non - stationarity removal is differencing up to some order. To achieve first order differencing from each observation the previous one is subtracted. The second order differencing is accomplished by repeating this process on obtained time series. It is similar for higher orders. Admittedly, these transformations can lead to the stationarity, but as a consequence all memory from the original series is erased. On the other hand the predictive power of machine learning algorithm is based on this memory. Lopez de Prado [2018] calls it as the stationarity versus memory dilemma by asking a question whether there is a trade-off between these two concepts. In other words – does exist a solution of making the time series stationary concurrently with keeping its predictive power. One way to resolve this dilemma – fractional differentiation - has been proposed by Hosking [1981]. Lopez de Prado upgraded this idea to find the optimal balance between zero differentiation and fully differentiated time series.

The remainder of the paper is organized as follows. Next section gives an overview of fractional differentiation introduced above. The data used in this research is presented in section 3 and the method applied on this data is described in Section 4. Section 5 discusses the results and chapter 6 concludes the paper.

2. Fractional differentiation – an overview

In this section the concept of fractional differentiation is elaborated in more detail. A real-valued feature can be expressed as the sum of all the past values. To each past value a weight ω_k is assigned:

$$\tilde{X}_t = \sum_{k=0}^{\infty} \omega_k X_{t-k}$$

The current value in time series is the function of all the past value occurred before this time point:

$$X = \{X_1, X_{t-1}, X_{t-2}, \dots, X_{t-k}, \dots\}$$

The application of fractional differentiation on time series allows to decide each weight for each corresponding value. To present it we can begin from a matrix of observations denoted as $\{X_t\}$ and the lag operator B, where:

$$B^k X_t = X_{t-k}$$

for $k \geq 0$ and $t > 1$. The differencing of first order can then be expressed as:

$$(I - B)X_t = X_t - BX_t = X_t - X_{t-1}$$

Knowing that:

$$(1 - B)^2 = 1 - 2B + B^2$$

$$B^2 X_t = X_{t-2}$$

we can derive the second order of differentiation:

$$(I - B)^2 X_t = X_t - 2X_{t-1} + X_{t-2}$$

Using binomial coefficients the series can be expanded to:

$$(I - B)^d = \sum_{k=0}^{\infty} (-B)^k \prod_{i=o}^{k-1} \frac{d-i}{k-i} = 1 - dB + \frac{d(d-1)}{2!}B^2 - \frac{d(d-1)(d-2)}{3!}B^3 + \cdots$$

because from the mathematical point of view:

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

where n is a positive integer. In addition we determined d as a real number and hence we received:

$$(1+x)^d = \sum_{k=0}^{\infty} \binom{d}{k} x^k$$

All weights were calculated by fractional derivative and can be expressed as:

$$\omega = \left\{1, -d, \frac{d(d-1)}{2!}, -\frac{d(d-1)(d-2)}{3!}, \dots, (-1)^k \prod_{i=0}^{k-1} \frac{d-i}{k!}, \dots\right\}$$

When we consider d as a positive integer number there is a point when d is equal to k, then $d-k=0$ and:

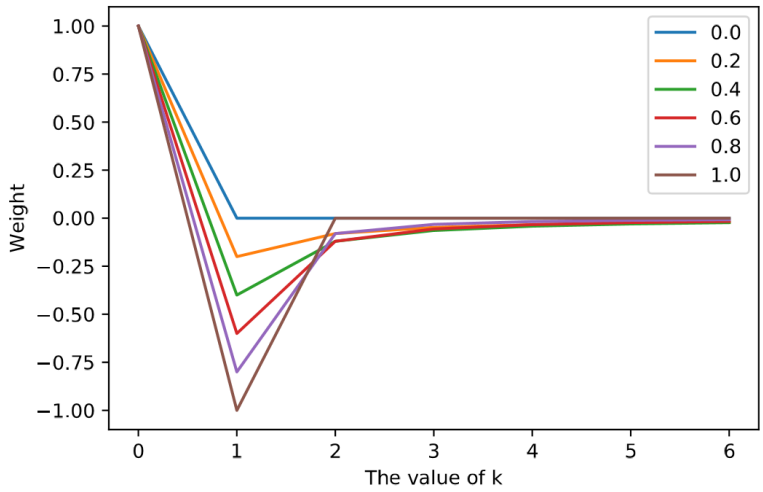
$$\prod_{i=0}^{k-1} \frac{d-i}{k!} = 0$$

Taking into consideration the information that dot product is zero we can conclude that memory beyond that point is removed. In first order differencing ($d=1$) weight follow as (see Figure 1 as a confirmation):

$$\omega = \{1, -1, 0, 0, \dots\}$$

General-purpose approach of coefficient for various orders of differencing presents Figure 1. For example, if $d = 0.25$ and k is always an integer number all weights achieved values other than 0 which means that the memory is going to be preserved.

Figure 1 Weights of the lag coefficients for various values of k



Notes: Each line as related to particular order of differencing ($d \in [0,1]$)

From the above derivation the iterative formula for the weights of the lags can be deduced:

$$\omega_k = -\omega_{k-1} \cdot \frac{(d-k+1)}{k}$$

where ω_k is the coefficient of backshift operator B^k. For the first order differentiation the following coefficient can be concluded (likewise weights mentioned before): $\omega_0 = 1$, $\omega_1 = -1$. $\omega_k = 0$ for $k > 1$.

In conclusion, the main intention to use the fractional differentiation is finding the fraction d, which is considered as minimum number needed to achieve stationarity, meanwhile keeping the maximum volume of memory in analyzed time series.

3. Data

This study uses four datasets with main stock indexes from different countries: WIG20 (Poland), S&P 500 (USA), DAX (Germany) and Nikkei 225 (Japan). The stock indexes were recorded from 1st June of 2010 to 30th June of 2020. The empirical distributions of mentioned indexes observed in each of the datasets are given in Figure 2.

Figure 2 Time series of selected stock indexes

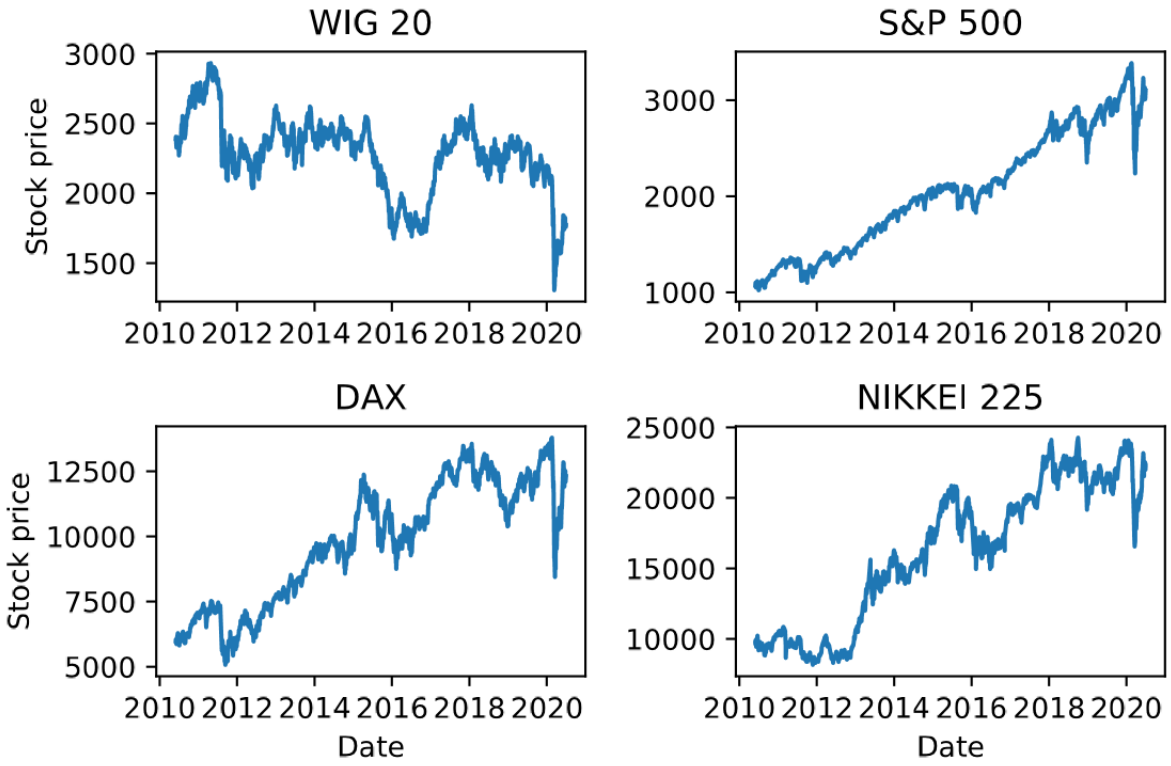


Table 1 shows the results of the unit root test for all the analyzed stocks, including the augmented Dickey-Fuller (ADF) test and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test. These tests are used to determine whether a given time series is stationary. The null hypothesis of the ADF test essentially assumes non-stationarity and the null hypothesis of the KPSS test is stationarity. It can be found out that all the stock prices are non-stationary.

Table 1 The results of ADF and KPSS tests

Stock index	ADF	KPSS
WIG 20	-2.22 (0.19)	2.64 (0.01)
S&P 500	-0.71 (0.85)	8.88 (0.01)
DAX	-1.61 (0.48)	8.12 (0.01)
NIKKEI 225	-1.21 (0.69)	8.18 (0.01)

Notes: In parentheses the p-value of tests is provided.

4. Method

This section describes the selected approach used to compare statistical properties of fractional differentiation with differencing of first order.

Studying the literature review proposed by Guresen et al. (2011) it can be concluded that artificial neural networks (ANN) were often implemented in forecasting stock prices indexes. In general, ANN outperform other statistical models applied on time series due to their good nonlinear approximation ability (Qiu et al. 2020). They were inspired by the strategy human brain processes given information. One of the most frequently implemented neural networks topology is multilayer perceptron.

Like the human brain the neural network has single processing elements which are called neurons. They are connected to each other by weighted and directed edges (see Figure 3). Commonly, neurons are aggregated into layers. Typical multilayer perceptron has three layers consisting of neurons: input layer, output layer and hidden layer. In the most simple case of artificial neural network, the edges between layers are limited to being forward edges (feed

forward artificial neural networks). It means that any element of a current layer feeds all the elements of the succeeding layer.

In the first step the system takes the values of neurons in the input layer and sums them up by the assigned weights. In this first iteration all weights are randomized. Then, for each iteration an error is calculated, which is a difference between the achieved value and the output value.

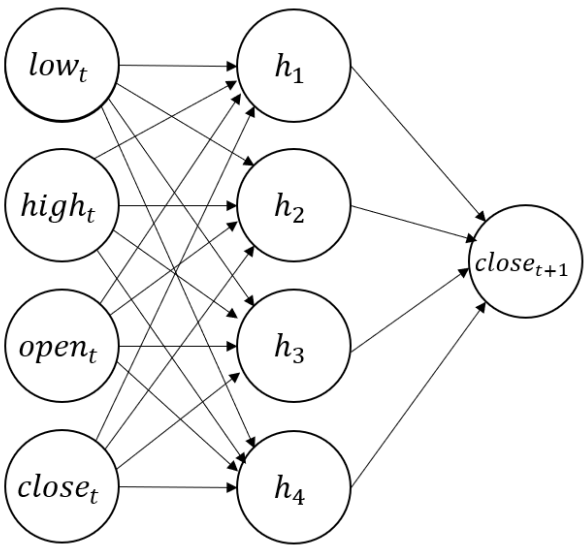
The goal of neural network is mapping values from input layer to values from output layer using hidden neurons in some way. This mapping is based on modifying the weights of the connections to receive a result closer to the output. To determine the value of the output applying the activation function to a weighted sum of incoming values is needed as well. The most widely used activation functions are: the logistic function and the hyperbolic tangent (Guresen et al. 2011). This learning process can be conducted with, for example, Levenberg-Marquardt backpropagation algorithm:

$$S(\beta) = \sum_{i=1}^{m} [y_i - f(x_i, \beta)]^2$$

where m is the size of time series. The sum $S(\beta)$ should be minimized.

There are other learning techniques used to train neural network models such as scaled conjugate gradient, one step secant, gradient descent with adaptive learning rate, gradient descent with momentum (Moghaddam et al. 2016).

Figure 3 An example of simple ANN with input, hidden and output layers



In this study we are focused on predicting the closing price of a stock tomorrow $\{close_{t+1}\}$ which is the output layer using the input layer consisting prices measured a day before $\{low_t, high_t, open_t, close_t\}$. The structure of this ANN is presented in Figure 3.

The performances of the resulting neural network are measured on the test set according to following metrics:

- root mean square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)}{N}}$$

- mean absolute error (MAE):

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

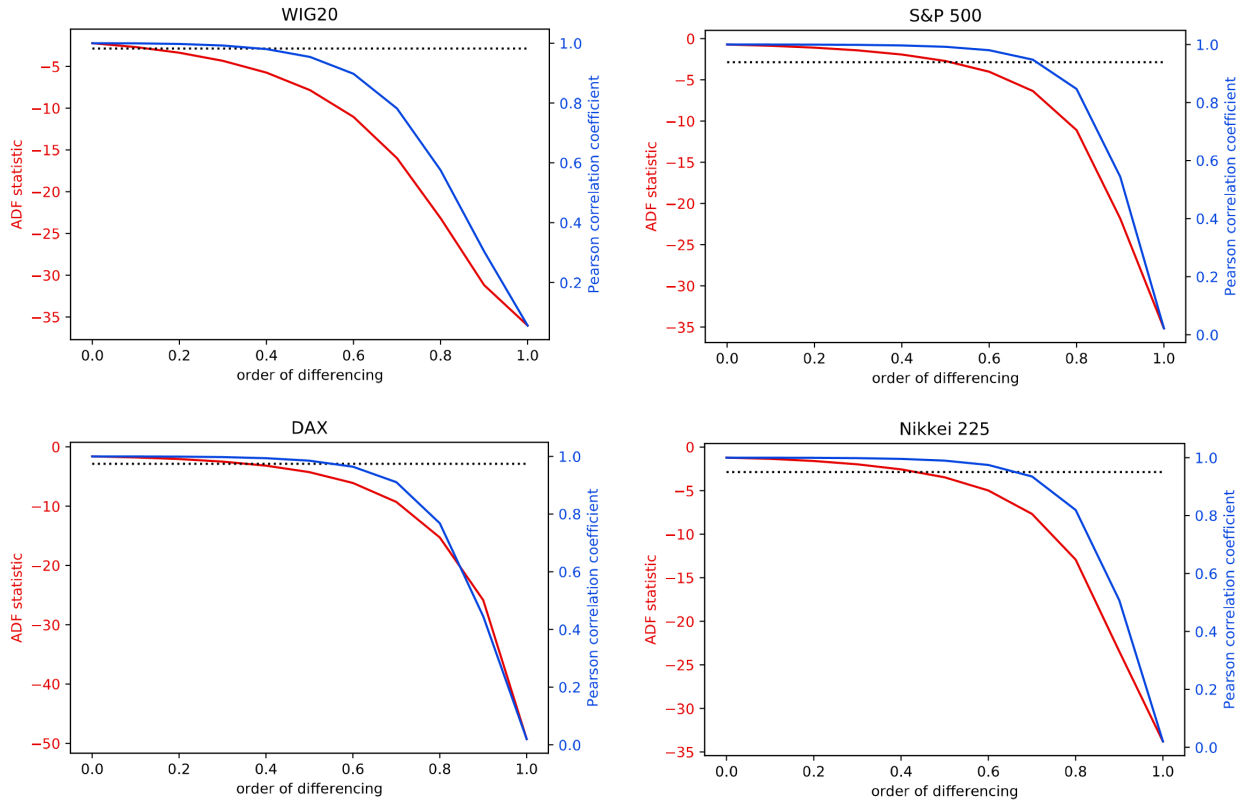
where N denotes the number of observations, $\hat{y}$ is the model prediction value and y_i is the observed value.

5. Results

In this section the fractional differentiation method will be applied on stock indexes described in previous part. For every stock index we are going to compute the minimum coefficient d to get stationary fractionally differentiated series.

To find the minimum coefficient d the combination of the Augmented Dickey-Fuller test statistics and Pearson correlation coefficient will be used. This concept is illustrated in the Figure 4 and Table 3 (Appendix A). The ADF statistic is on the left y – axis, with the correlation between the original series and the fractionally differenced series on the right y – axis.

Figure 4 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to selected stock indexes.



The original series of WIG20 index has the ADF statistics of -2.22, while the differentiated equivalent has this statistic equal to -36.02. At a 95% confidence level, the critical value of the DF t-distribution is -2.8623. This value is presented as a dotted line in Figure 4. The ADF statistic crosses this threshold in the area close to $d = 0.1$. At this point the correlation has the high value of 0.9975. This proves that fractionally differenced series is not only stationary but holds considerable memory of the original series as well.

Similarly, the ADF statistic S&P 500 index reaches 95% critical value when the differencing is approximately 0.4 (for DAX series $d \approx 0.3$ and Nikkei 225 $d \approx 0.4$) and the correlation between the original series and the new fractionally differenced series is over 99% (the same for DAX and Nikkei 225).

Figures 5-8 contain original series with results of implementing the minimum coefficient d indicated above. The high correlation indicates that the fractionally differenced time series retains meaningful memory of the original series.

Figure 5 WIG20 stock index (in blue, left axis) along with fractional derivatives (shades of green).

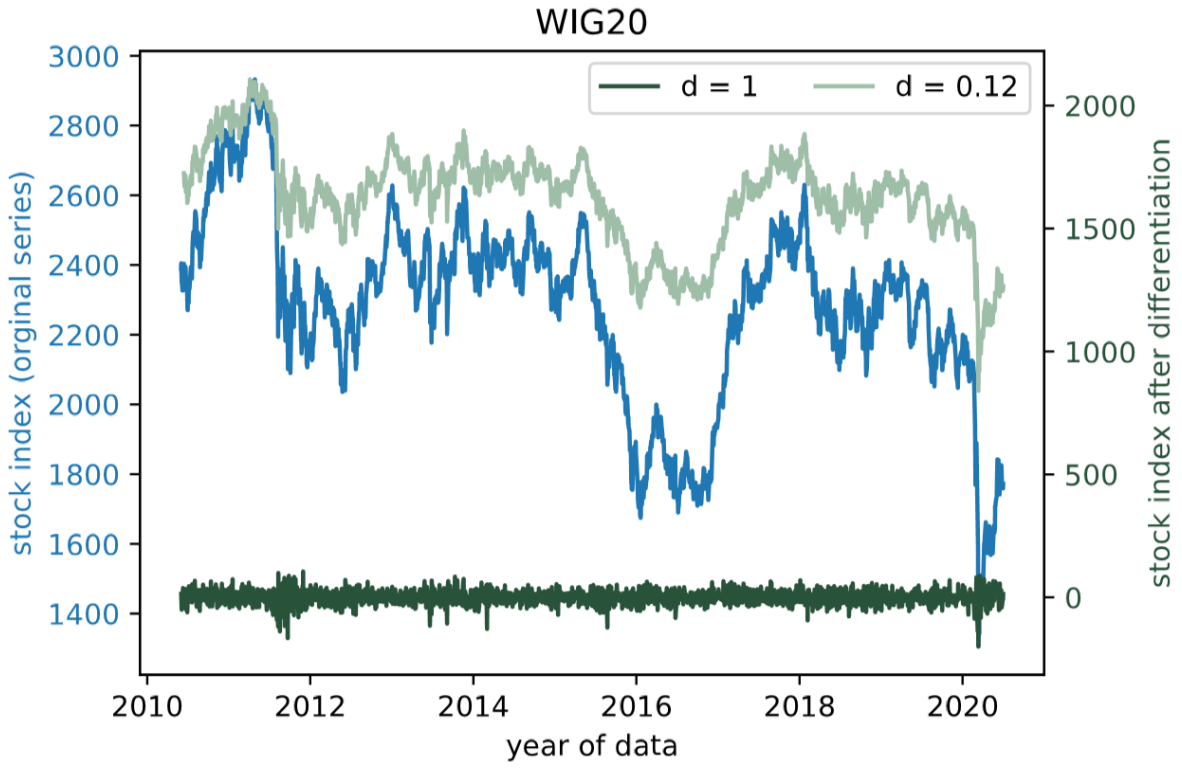


Figure 6 S&P 500 stock index (in blue, left axis) along with fractional derivatives (shades of green).

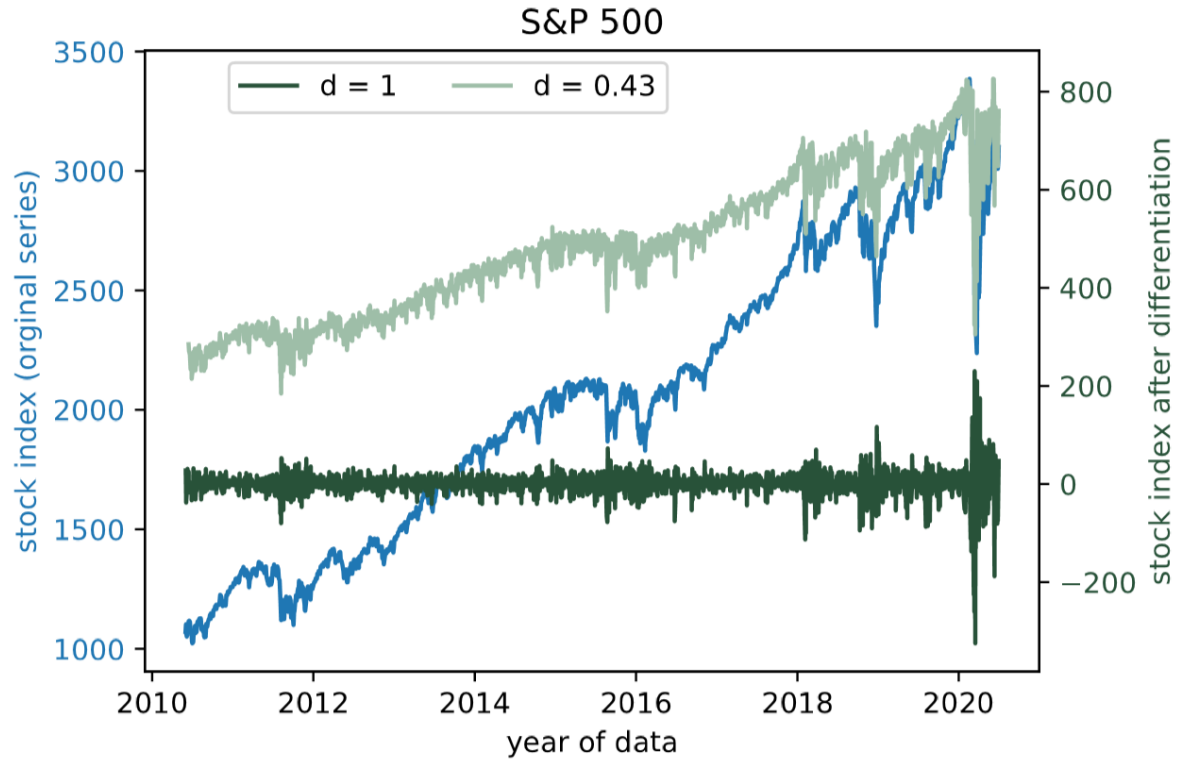


Figure 7 DAX stock index (in blue, left axis) along with fractional derivatives (shades of green).

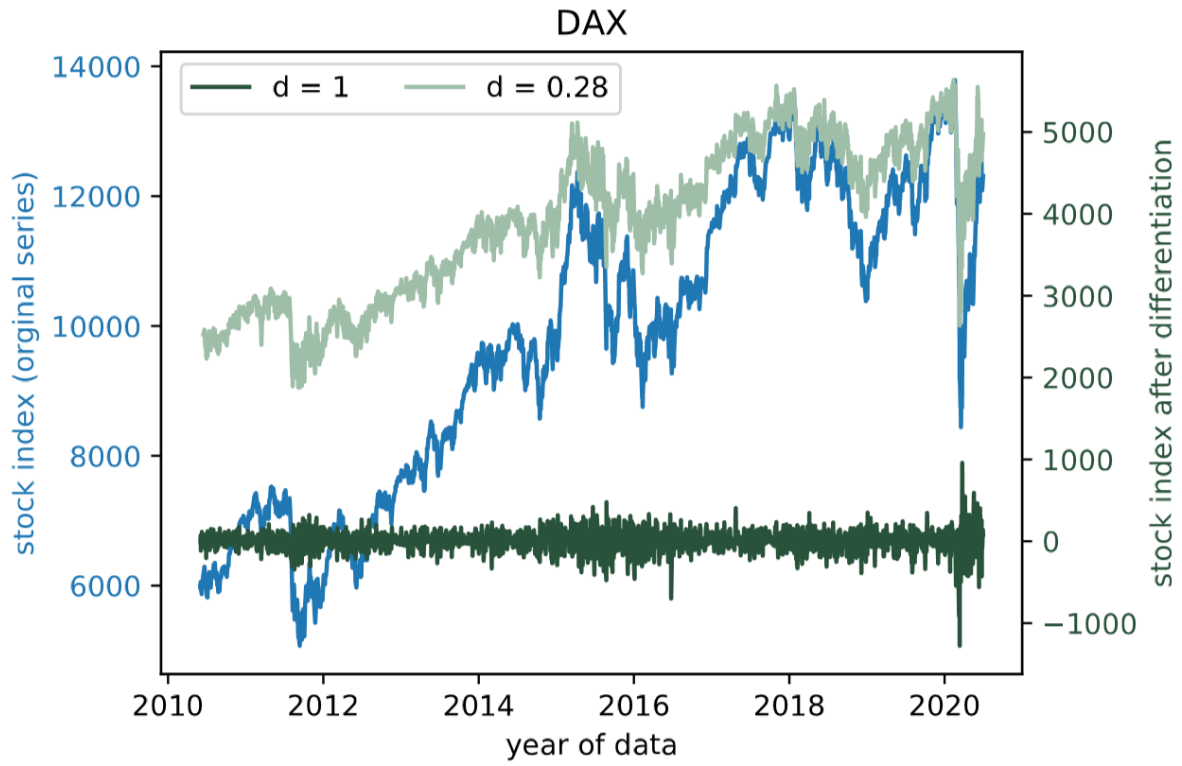
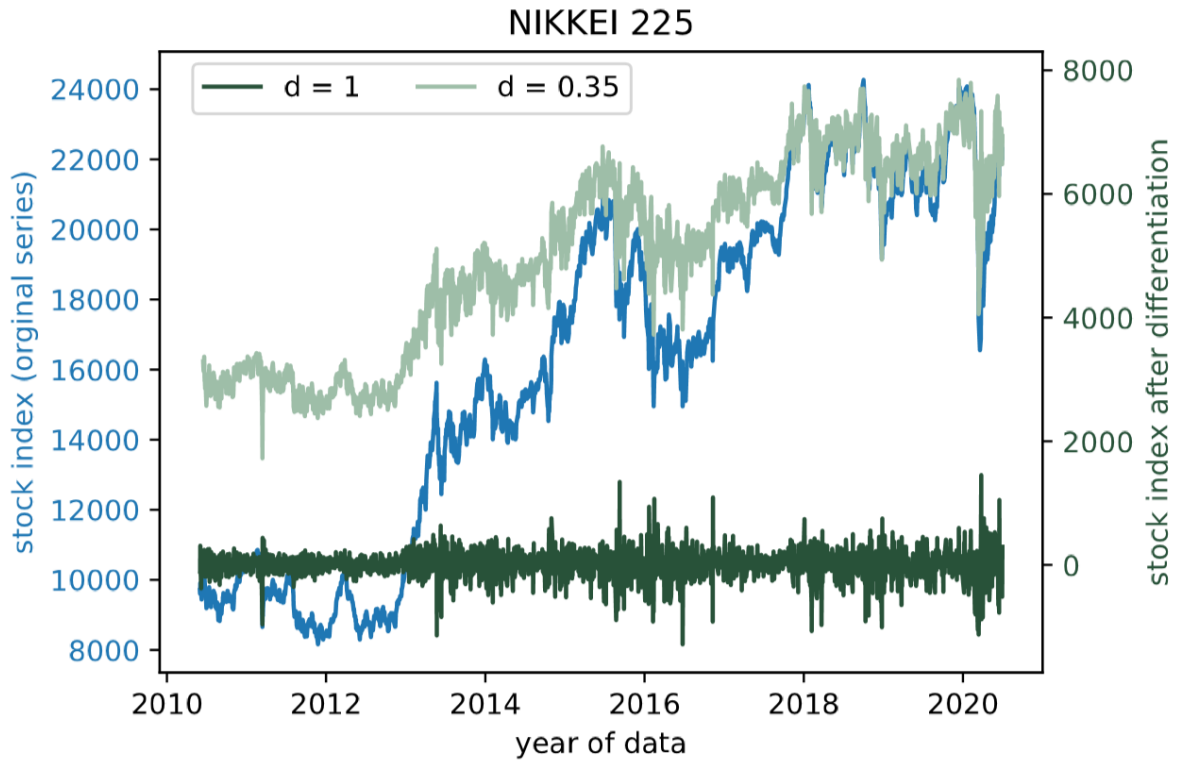


Figure 8 Nikkei 225 stock index (in blue, left axis) along with fractional derivatives (shades of green).



Above obtained time series are going to be implemented in creating multiplayer perceptrons for proposed stock indexes. To begin with, the data for all stock indexes has been normalized using the following equation:

$$price_{norm} = \frac{price - \min(price)}{\max(price) - \min(price)}$$

and divided into training and testing datasets. First 1681 days are used for training and last 813 used for testing process.

A feedforward neural networks were created using Keras, open-source neural-network library in Python. Every network were inputted with the low, high, opening and closing price for each day t. The result layer consists of closing price on next day $t + 1$:

$$\begin{bmatrix} low_t \\ high_t \\ open_t \\ close_t \end{bmatrix} \rightarrow [close_{t+1}]$$

It means that artificial neural network predicts the closing price of the next days using historical data from the day before.

As it is observed in Table 2, the analysis shows that for all stock indexes fractional differentiation gives better RMSE and MAE statistics obtained on test data.

Table 2 Results of ANN on test datasets

Stock index		**RMSE**	**MAE**
WIG 20	d = 0.12	18.10	13.68
	d = 1	27.31	20.37
S&P 500	d = 0.43	36.32	20.64
	d = 1	41,10	29.38
DAX	d = 0.28	124.07	84.43
	d = 1	142.45	94.79
NIKKEI 225	d = 0.35	286.86	211.96
	d = 1	317.57	249.80

The purpose of this research is not to evaluate the predictive performance of artificial neural networks, but rather to evaluate how much better a fractional differentiation is, compared to full differentiation.

6. Conclusions

In this study the concept of fractional differentiation was evaluated on four time series datasets obtained from the known stock exchanges (from Poland, United Kingdom, Germany and Japan). In these fractionally differenced time series selected orders of differencing vary from 0.12 to 0.43, which is far from integer differencing. For all of them we have received a high level of linear correlation coefficients (above 0.99%), which means immense association with original series. Nonetheless, these fractional time series are stationary (indicated by the results of Augmented Dickey-Fuller test), which proves that their means, variances and covariances are time-invariant.

Using fractional differentiation we have made analyzed time series stationary while keeping its memory and predictive power. Therefore, this study has clearly demonstrated the potential of applying fractional differentiation on time series.

The previously discussed results clearly show the benefit of fractional differentiation compared to classical differentiation in terms of applied performance measurements on created artificial neural networks.

7. References

Bishop, C., 1995. Neural network for pattern recognition. Oxford University Press.

Guresen, E., Kayakutlu, G., Daim, T. U., 2011. Using artificial neural network models in stock market index prediction. Expert Systems with Applications 38, 10389-10397.

Hearty, J., 2016. Advanced Machine Learning with Python. Packt Publishing.

Hosking, J., 1981. Fractional differencing. Biometrika 68 (1), 165-175.

Lopez de Prado, M. 2018. Advances in Financial Machine Learning, in: Lopez de Prado, M., Advances in Financial Machine Learning. John Wiley&Sons, Inc., Hoboken, New Jersey, 75-90.

Medeiros, M. C., Teräsvirta, T., Rech, G., 2005. Building neural network models for time series: a statistical approach. Journal of Forecasting 25, 49-75.

Moghaddama, A. H., Moghaddamb, M. H., Esfandyari, M., 2016. Stock market index prediction using artificial neural network. Journal of Economics, Finance and Administrative Science 21, 89-93.

Principe, J. C., Euliano, N. R., Lefebvre W. C., 1999. Neural and Adaptive Systems: Fundamentals through Simulations. John Wiley&Sons.

Qiu, J., Wang, B., Zhou, C., 2020. Forecasting stock prices with long-short term memory neural network based on attention mechanism. PLoS ONE 15, 1-15.

White, H., 1990. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. Neural Networks 3, 535-549.

Zhu, X., Wang, H., Xu, L., Li, H., 2008. Predicting stock index increments by neural networks: The role of trading volume under different horizons. Expert Systems With Applications 34, 3043-3054.

Appendix A

Table 3 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to selected stock indexes

Order of differencing	WIG 20		S&P 500		DAX		Nikkei 225	
	ADF	CORR	ADF	CORR	ADF	CORR	ADF	CORR
0.0	-2.22	1.0000	-0.71	1.0000	-1.61	1.0000	-1.21	1.0000
0.1	-2.69	0.9995	-0.87	0.9999	-1.78	0.9999	-1.34	0.9999
0.2	-3.36	0.9975	-1.09	0.9996	-2.05	0.9992	-1.59	0.9995
0.3	-4.33	0.9922	-1.43	0.9988	-2.49	0.9976	-1.97	0.9984
0.4	-5.74	0.9803	-1.93	0.9968	-3.17	0.9939	-2.55	0.9958
0.5	-7.84	0.9542	-2.71	0.9921	-4.27	0.9852	-3.47	0.9897
0.6	-11.04	0.8978	-4.02	0.9804	-6.09	0.9643	-4.98	0.9748
0.7	-15.99	0.7817	-6.36	0.9477	-9.29	0.9107	-7.67	0.9351
0.8	-23.17	0.5755	-11.10	0.8468	-15.28	0.7681	-12.92	0.8194
0.9	-31.17	0.3040	-21.84	0.5441	-25.85	0.4454	-23.46	0.5074
1.0	-36.02	0.0560	-35.16	0.0220	-49.28	0.0199	-33.79	0.0202

University of Warsaw
Faculty of Economic Sciences
44/50 Długa St.
00-241 Warsaw
www.wne.uw.edu.pl