Alishev Elgun Rafiq, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 12, Issue 8, August 2022, pp. 05-10

RESEARCH ARTICLE OPEN ACCESS

Development of a Neural Network for Prediction of Financial Conditions (Shares and Their Prices) To Make Important Investment Decisions

SeyidovaIradaBahadur
Azerbaijan State Oil and Industry University
Baku, Azerbaijan
e-mail: irada_seyid@mail.ru

Alishev Elgun Rafiq
Azerbaijan State Oil and Industry University
Baku, Azerbaijan
e-mail:ealishev43@gmail.com

Annotation. The purpose of this article is to demonstrate how to transform non-stationary data into stationary data using the fractional differentiation method, and then compare it with other methods. Next, create the neural network model itself, suitable for our data, which will be able to predict the trend of stocks with high probability, as well as predict stock prices for tomorrow with approximate accuracy, while helping in making important investment decisions.

Keywords: Programming languages, Recurrent neural networks, LSTM, GRU, Bidirectional Neural networks, Stationary Data, EDA, Fractional differencing.

Date of Submission: 24-07-2022 Date of Acceptance: 06-08-2022

I. Introduction

In the era of the digital industry in any field, the use of modern information systems is required to ensure high speed of work, the choice of accurate solutions, and the minimization of possible errors. One such area is the financial and economic area, where there is a great demand for forecasting tools. In this regard, this article discusses obtaining stationary time series using fractional differentiation, as well as the development of a neural network that predicts the obtained time series (stocks and their prices) for making important investment decisions.

Today, the ability of a neural network, in particular recurrent neural networks, to predict, directly follows from its ability to generalize and highlight hidden dependencies between input and output data. And since, after training, the network is able to predict the future value of a certain sequence based on several previous values or some currently existing factors, it should be noted that forecasting is possible only when previous changes really predetermine future ones to some extent.

To model a neural network, we need to use some kind of development environment. One of the main stages is the choice of a programming language in which the neural network model will be created. The Python programming language was chosen for development.

Formulation of the problem. According to research, in order for a neural network model to accurately predict, the condition of time series stationarity must be met, and the right model must be selected based on the study of time-series data. The removal of memory-storing data by detrending and differentiating with integers is very likely to produce forecasts that are inaccurate. The study also demonstrated that many economists have trouble using fractional differentiation to transform non-stationary data into stationary data. After evaluating other methodologies, the goal of this thesis is to develop an automated procedure for employing the fractional differentiation method to convert non-stationary data into stationary data. Then creation of a neural network model that is appropriate for our data and capable of accurately forecasting tomorrow's stock prices as well as the trend of stocks, which will be used to guide crucial investment decisions.

Solution of the problem. To solve this problem, we will use the Keras library package. Keras is a high-level, open-source framework created by GOOGLE that is written in Python. This framework is easy to

use and also makes the process of creating a neural network more convenient. Additionally, this framework has ready-made model types such as: LSTM, GRU and others. But the main feature of this framework is its speed. The Keras framework allows you to change the backends that are the core of this framework, contributing to speed. There are various backends such as: Tensorflow, Theano, PlaidML, MXNet, Pythoch. In our case, the Tensorflow backend was chosen. Tensorflow is an open source machine learning and artificial intelligence development library developed by GOOGLE. The advantage of this library is that this library is developed in the low-level C++ language, which greatly increases the speed of calculations.It can be used to solve various problems, but a special place is occupied by the training and inference of deep neural networks. Additionally, this library is cross-platform, which makes it one of the most popular libraries in the world of information technology. Also, we need the open source Numpy and Pandas modules. Before starting to create a model, we will need to perform some operations on time series. The data is large, so the calculation can take a long time. To avoid these situations, we can use previously written modules.Numpy is an open source module that provides compiled fast functions that provide mathematical and numerical operations written in the low-level C language for high performance. This module provides the user with basic methods for manipulating arrays and matrices. Pandas is an open source library that works in conjunction with the Numpy module. This library is mainly needed for data manipulation and analysisA unique data structure like Series or DataFrame is this library's key feature.In particular, these structures allow for faster and more convenient manipulations and analyses with numerical tables and time series. Additionally, we will need libraries to provide results using graphs, for this the matplotlib library will be used. Matplotlib is an open source library for displaying the results of Numpy's numerical math extensions.

The Scikit-learn library will be used to evaluate the results of the models. Scikit-learn is a free machine learning library that includes various classification, regression, and clustering algorithms, as well as metrics for model evaluation. You also need an environment to implement all of the above. This article uses the Jupyter Notebook development environment. Jupyter Notebook is a free interactive development environment in the form of a web application. The user has the ability to break the code into small parts and work on them.Additionally, it is possible to work on functions in any order. A feature of this web application is the interactivity of the work environment, you can write code and immediately see the result. It has a simplified shell of work, as well as a flexible choice of the kernel in accordance with the user's requirements. This interactivity of the web application is very convenient for data analysis, whereby changing the parameters of the function, you can immediately visualize the result that is easy for the user to understand. Additionally, there is the possibility of direct control of operations.or a deeper study, several models were created. We also tested two stationary time series obtained by the fractional differentiation method, however, one of them was also passed through the logarithm function. Based on the evaluation of several neural network models, as well as two stationary time series, fractional differentiated data by the logarithm function were selected to predict the neural network model. There are several reasons for this choice:The fractionally differentiated logarithm function data varies between smaller values, hence the data fluctuates less than the fractionally differentiated data.Fractionally differentiated data by the logarithm function have a smaller order of difference, therefore, they have more memory relative to fractionally differentiated data.The next step is to test and evaluate different models. It is necessary to determine the number of epochs, the batch size after which the weight values will change, the number of layers and neurons, the neural network model (GRU, LSTM, Bidirectional). To determine which one predicts the data more accurately, we will need to evaluate each model. We will evaluate the models according to the following metrics: MSE, RMSE, R-Squared, Adjusted R-Squared. Let's consider each of them separately. MSE (Mean Squared Error) indicates how close the regression line is to a set of points. Calculated using the following formula:

$$MSE = \frac{1}{n}\sum_{1}^{n}(Y_i - \hat{Y}_i)^2 \qquad (1)$$

where, n is the amount of data, $Y_i$is the real data, $\hat{Y}_i$are the predicted results. RMSE (Root Mean Squared Error). standard deviation of residuals (forecast errors). In other words, it tells you how concentrated the data is around the line of best fit. The RMSE is a more easily demonstrative statistic because it has the same units as the quantity on the y-axis. It is calculated using the following formula:

$$RMSE = \sqrt{MSE} \qquad (2)$$

R-Squared (R2, R-Squared) is a measure often used to evaluate linear regression, always between 0 and 100. Shows how close the data is to the regression, also known as the coefficient of determination. Answers the question whether the model explains well the changes in the dependent variable. Calculated using the following formula:

$$R2 = 1 - \frac{\sum(\hat{y}-y_i)^2}{\sum(y_i-\bar{y})^2} \quad (3)$$

where, y_i are the actual values of Y for each observation, $\hat{y}$ are the values predicted by the model, $\bar{y}$ is the average of all y values. Consider also the adjusted R2. Adjusted R2 is the adjusted measure of fit for linear models. However, there is one major difference between R2 and adjusted R2: R2 assumes that each individual variable explains the change in the dependent variable. Adjusted R2 shows the percentage of variation explained only by the independent variables that actually affect the dependent variable.However, there is one major difference between R2 and adjusted R2: R2 assumes that each single variable explains the change in the dependent variable. The adjusted R2 shows the percent variation explained only by the independent variables that actually affect the dependent variable. The adjusted R2 is calculated using the following formula:

$$Adj\ R2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1} \quad (4)$$

where, n is the size of common samples, p is the number of predictors. After the evaluation measures of the model are defined, we need to create various variants of models and types, evaluate them and choose the optimal one. It should also be noted that the program is written in such a way that when retraining or prediction worsens, the program interrupts the learning cycle on its own and saves the best values of the weights. In other words, for each model, we can write 300 epochs, where the program, when the results deteriorate, calls Callback and stops the training cycle. Also based on the MSE metric values of the test data, updates the model weights if the metric values are better than the previous one. Thus, even when retraining the model, we will have the best weight values. Considertable 1.

Metric values MSE
tab 1

Modelname	**Modellayers**	
	Layertype	**Size**
Model_1	GRU	12
	Dropout (0.2)	12
	GRU	8
	Dropout (0.1)	8
	Dense	1
Model_2	LSTM	30
	Dropout (0.1)	30
	LSTM	20
	Dropout (0.1)	20
	Dense	5
	Dense	1
Model_3	Bidirectional (LSTM)	24
	Dropout (0.2)	24
	Bidirectional (LSTM)	12
	Dropout (0.1)	12
	Dense	1
Model_4	Bidirectional (GRU)	24
	Dropout (0.2)	24
	Bidirectional (GRU)	12
	Dropout (0.1)	12
	Dense	1
Model_5	GRU	12
	Dropout (0.2)	12
	LSTM	6
	Dropout (0.1)	6
	Dense	1

This table (tab.1) shows 5 models that showed relatively good results. The following table (tab.2) shows the results of evaluating model prediction on test data, and records the data on the measurements selected earlier.

PredictionEvaluationResults

tab.2

Modelnames	Numberofparties	Numberofepochs	MSE	RMSE	R2	Adj R2
Model_1	5	17	0.217	0.465	0.995	0.994
Model_2	12	23	0.487	0.698	0.988	0.988
Model_3	6	28	1.945	1.394	0.955	0.954
Model_4	6	18	0.362	0.602	0.991	0.991
Model_5	6	22	0.356	0.596	0.991	0.991

After analyzing the table (Table 2), you can see that the Model_1 model, which consists of layers of the GRU model, has the best indicator. The second good result is obtained by the mixed model Model_5. Followed by Model_4, Model_3 respectively. Finally, the bidirectional LSTM model has the weakest data among the models.Therefore, it can be seen that in this type of data, models of the GRU type are more or less a good choice. This may be due to the fact that GRU models give better results on small data than LSTM models. It can also be seen when looking at the training plots based on the MSE values.

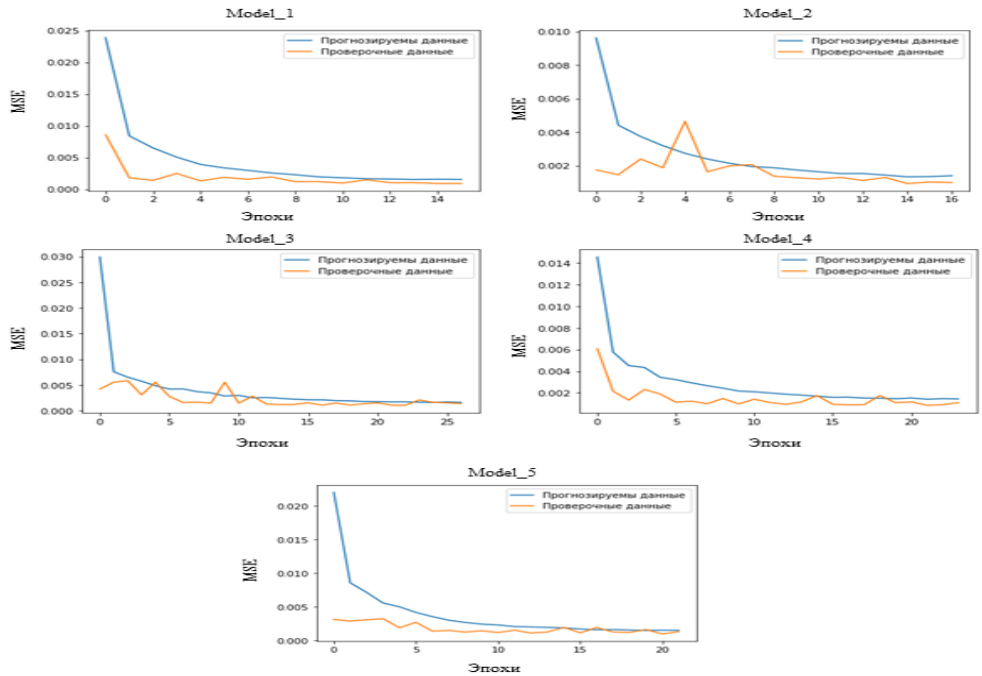
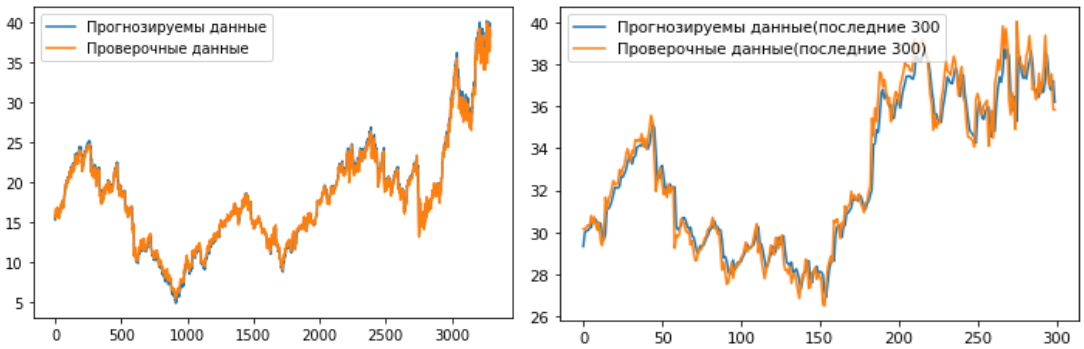


Fig. 1

The graph (fig. 1) shows the dependence of the MSE value on the number of epochs. If you look closely, you can see that GRU models have smoother learning transitions than LSTM models. This fact strengthens our assumptions. Summing up all the above conclusions, we get that the most suitable model is Model_1, therefore, the creation of a neural network model will be based on Model_1, consisting mainly of layers of the GRU type. For time series forecasting, the Model_1 model was chosen, which showed good results compared to other models. The next step is to create a model that will predict our data.After all these operations, at the end we get a model that can predict financial time series. Consider the graph (Fig.2(A, B))



A) **B)**

Fig.2(A,B)

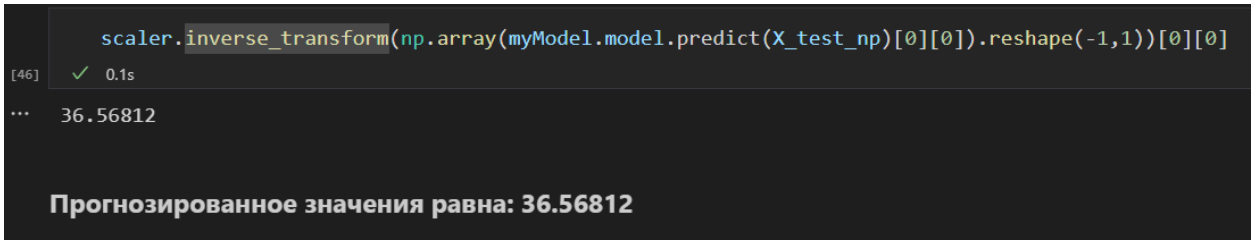
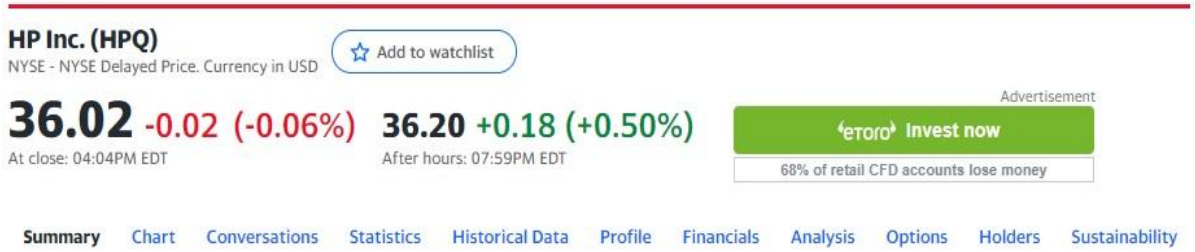


Fig.3

Let's consider the real values for the same period, which is shown in the figure (Fig.3)



II. CONCLUSION

In this paper, we solved the problem of predicting the movement of HP Inc. closing prices. using neural networks. As shown by the results. neural networks can predict closing price movements with high probability, provided there is no additional information about the factors that can affect stock price movements. Based on the results obtained, we can conclude that in order to solve the problems of predicting the closing prices of company shares, it is possible to train a neural network to analyze and derive a forecast of stock prices with a high degree of probability.But for more accurate forecasting, it is necessary to train the model on stationary data. The test results showed that when training a neural network on stationary data, the model produces more accurate predictions. In stationary data, there is no trend, no seasonality, which makes it possible for the neural network model to predict with great accuracy in the absence of additional factors affecting the forecasting of the neural network. In addition, based on all the findings of the research and test results, it can be concluded that in a world of rapidly developing digital industry, neural networks can help the decision maker make more accurate decisions based on the data predicted by the neural network.

LIST OF REFERENCES

[1]. GrzegorzDudek, SlawekSmyl, PawelPelka "Recurrent Neural Networks for Forecasting Time Series with Multiple Seasonality: A Comparative Study". 2022
[2]. Fan Chen "Deep Neural Network Model Forecasting for Financial and Economic Market". 2021
[3]. RafałWalasek, JanuszGajda, "Fractional differentiation and its use in machine learning". 2021

[4]. Luca Barbagliaa, Sergio Consolia ,SebastianoManzanb, "Forecasting with Economic News". 2022
[5]. AnandaChatterjeeHrisavBhowmick, Jaydip Sen, "Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models".] 2019
[6]. Sidra Mehtab and Jaydip Sen, "A Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models". 2019
[7]. Swaroop Mishra, AnjanaArunkumar, "A Proposal to Study "Is High Quality Data All We Need?2018
[8]. Yuxuan Huang, Luiz Fernando Capretz, Luiz Fernando Capretz, "Machine Learning for Stock Prediction Based on Fundamental Analysis". 2021

SeyidovaIradaBakhadurovna, et. al. "Development of a Neural Network for Prediction of Financial Conditions (Shares and Their Prices) To Make Important Investment Decisions." *International Journal of Engineering Research and Applications (IJERA*), vol.12 (08), 2022, pp 05-10.