**Research on the effectiveness of**

**the applicability of neural networks**

**for time series forecasting**

Arkadiy Sergeevich Khorunzhenko1, Vasily Leonidovich Makushev2

**Abstract**

Time series forecasting plays a vital role in various domains, such as environmental analysis, weather forecasting, stock market predictions, and demand forecasting. This research paper investigates the effectiveness of neural networks for time series forecasting and compares them with standard econometric methods. The study also includes a comparison of different types of neural networks and their performance on datasets with varying frequencies. While forecasting does not provide precise outcomes, it can provide valuable insights into the direction of value movements, aiding decision-making in numerous applications.

Custom models are developed to address the specific requirements of each task, as generic models may not perform optimally on different datasets. This study aims to investigate the feasibility of using neural networks for time series forecasting of a financial data by implementing various models using the Python programming language. The models' performance is evaluated using such metrics as Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE), which provide a quantitative assessment of their accuracy. The implementation of neural network models was achieved using the PyTorch library.

# 1 Introduction

Time series forecasting is one of the most important modern tools for research and analysis in many fields. Time series are encountered in studies related to environmental analysis, such as forecasting the percentage of carbon dioxide in a region, weather, stock prices and indices, purchasing needs, and more.

Many tasks that have sufficient time series data can be automated and optimized through forecasting and its analysis. Although forecasting does not provide an exact result, it can indicate the direction of value movement, which can already help in many areas.

The relevance of developing custom models is dictated by the fact that each model is usually built for a specific task and does not always work on another dataset.

The aim of this work is to familiarize with methods of time series forecasting and to investigate the applicability of neural networks for time series forecasting. The implementation of custom models is dictated by the need to create a model base on a specific dataset for further analysis. Additionally, finding correctly functioning solutions for a given dataset is challenging, as there are many different datasets that differ not only in logical dependencies but also in data density.

In this work, various models were implemented using the Python programming language. For result verification, a number of metrics were selected, including MSE and MAPE. The PyTorch library was used for the implementation of neural network models.

Scientific literature presented in the list of references [1-8], as well as materials from scientific conferences and data from periodicals were used in the execution of this thesis.

# 2 Finding and selecting data

It was decided that for the accuracy of the study, it would be correct to choose a time period with a minimum number of global problems, since the initial models for forecasting will depend only on the data of the opening and closing prices. Therefore, it was decided to choose a period of time from 2010 to 2014.

Next, several foreign stocks and indices were selected, and among them there was a selection by values that had an increase in the values of this financial instrument over the period from 2010 to 2014.

Based on the results of all selection work, futures for the S&P500 and NASDAQ indices were selected. The data was downloaded from the site finam[1, 2] with a period of 1 minute. Further, if necessary, it will be possible to increase the period of data.

When getting acquainted with the models, we used S&P500 data with periods of a minute, 5 minutes, an hour, a day, and a week.

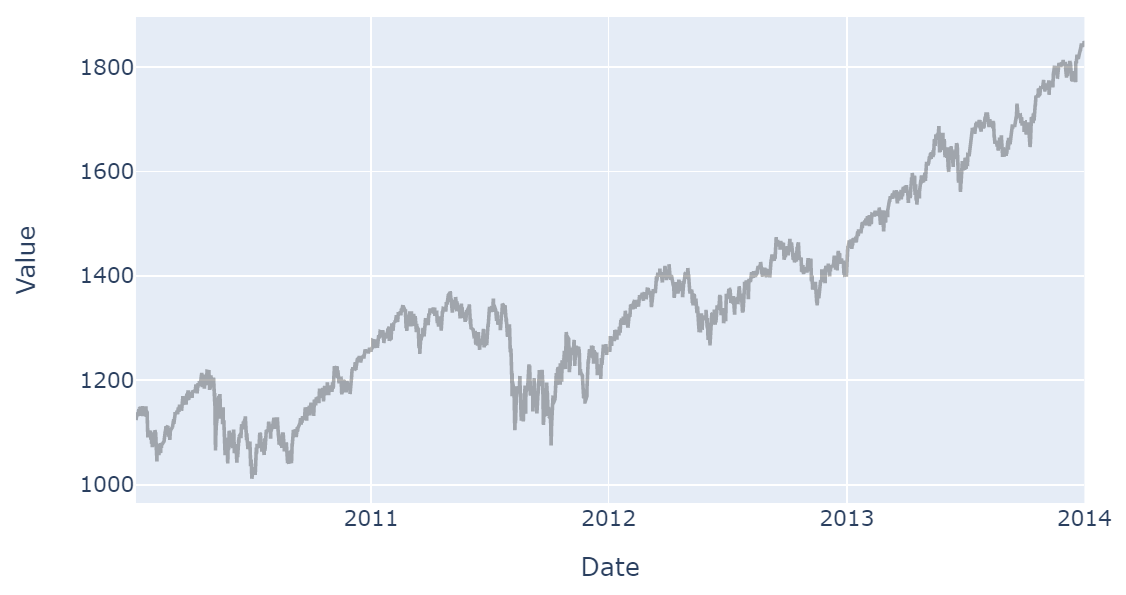


Figure 1. Chart of S&P500 index futures for the period from 2010 to 2014

# 3 Working with data

The present study utilized data known as time series. Time series data refers to information associated with specific moments or periods of time. It can include dates, times, durations, frequencies, and so on. Time series data can be represented in various formats, such as numbers, strings, or date and time objects. To work with time series data, it is necessary to identify the components comprising the given time series. Typically, a time series is represented in the following format

, (1)

where T(t) stands for a trend, a smooth change in the trend level; S(t) - stands for seasonality, periodic changes in the level of the series; C(t) - stands for cyclicity, changes in the level of a series with a variable period; E(t) is a random error, an unpredictable random component of the series.

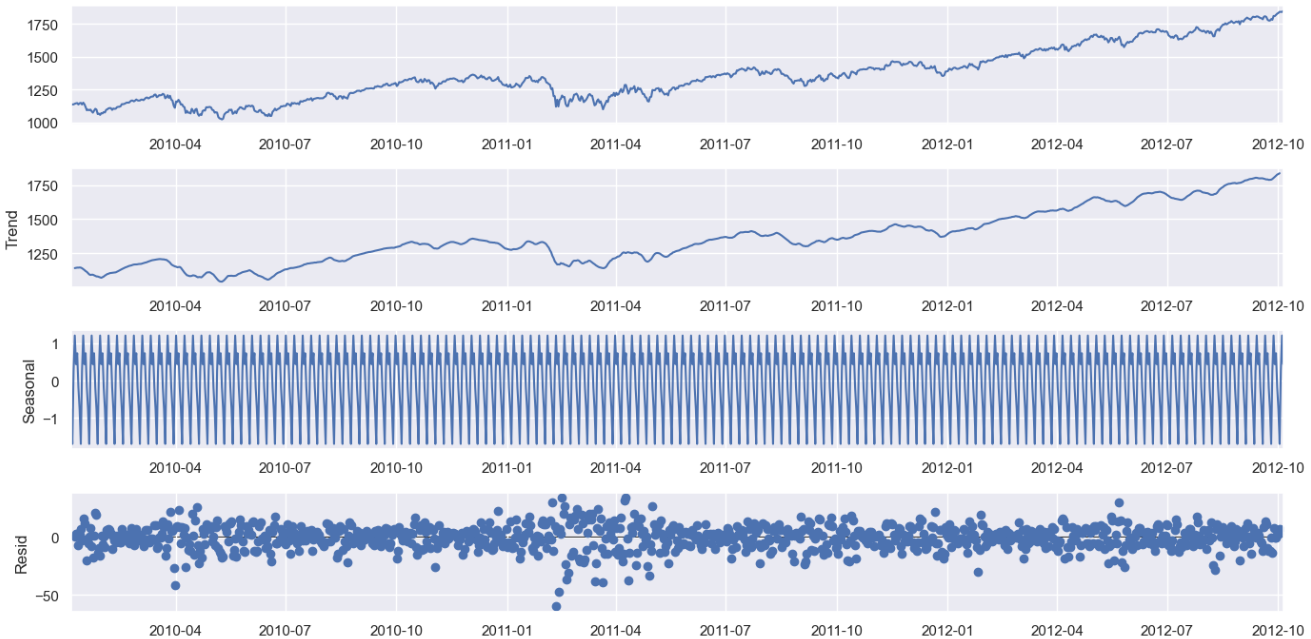


Figure 2. Time series, trend, seasonality and difference chart for the S&P500 2010-2012

To start time series analysis, first visualize the time series and look at trend, seasonality, difference, cyclicity, and random error graphs.

After a successful review of the trend, cycle and seasonal fluctuations, the time series is tested for stationarity. A time series is considered stationary if its statistical characteristics (mean, variance, etc.) do not change over time. To test the time series for stationarity, such methods as the Dickey-Fuller test, the Kwiatkowski-Phillips-Schmidt-Sheen test (KPSS) and the Philips-Perron test are used. If the series is not stationary, then various methods of data transformation should be applied, such as the first difference.

If the problem with stationarity has been solved, then further it is necessary to check the correlation analysis. In time series analysis, autocorrelation is often used, which measures the relationship between the values of a series at different points in time. This helps determine if the series has any dependencies on previous values.

After all the manipulations mentioned above, you can move on to creating a suitable model and forecasting the time series.

# 4 Statistical Models

A fairly large number of models were considered, ranging from the simplest ones to models that were specifically designed for time series forecasting. In this work, it was planned to consider the following models: AR, ARMA, ARIMA, ARCH, GARCH.

# 4.1 AR

The abbreviation AR stands for autoregressive model. This model is a time series prediction model in which the values of a time series currently depend linearly on the previous values of that series. The autoregression model can be written in the following form

where are autoregression coefficients, is a constant, is white noise.

The autoregressive model indicates that the output variable depends linearly on its previous values and on the stochastic term. This model is not always stationary, since it may contain a unit root.

# 4.2 ARMA

The autoregressive moving average model is already more advanced than the simple autoregressive model and generalizes the autoregressive model and the moving average model. The ARMA model is given by the following formula

where , are the autoregression and moving average coefficients, respectively, is a constant, is white noise.

To build a model, it is necessary to determine the order of the model, that is, the values of *p* and *q*, and then the coefficients themselves. To determine the coefficients, the maximum likelihood method or the least squares method is usually used.

# 4.2 ARIMA

The integrated autoregressive moving average model is a continuation of the ARMA(p,q) model. By definition, it turns out that the ARIMA(autoregressive integrated moving average) model is the difference of a time series of order d obeying the ARMA model.

The ARIMA(p,d,q) model for a non-stationary time series has the following formula

where is the time series difference operator of order d, the other parameters have the same meanings as in the explanation of the ARMA model.

In the future, it is also planned to conduct a study related to SARIMAX.

# 4.3 ARCH

This stochastic model is also autoregressive, but already has conditional heteroscedasticity. The ARCH model is appropriate when the error variance in the time series matches the autoregressive model, if an autoregressive moving average is assumed for the error variance[3], the model is a generalized autoregressive model with conditional heteroscedasticity[4]. This model is often used for the analysis of time series, with the help of which, in particular, financial mathematical time series can be described with non-constant volatility.

To model a time series using the ARCH model, it is necessary to enter a series of independent identically distributed random variables E, . A time series is called an ARCH(p) time series if it is recursively defined according to the following conditions[3]

where are real, non-negative parameters.

# 4.4 GARCH and its modifications

The idea of the ARCH model has been further developed in various directions, and today it is taken for granted as one of the advanced methods of econometrics.

Compared to ARCH, GARCH's conditional variance depends not only on the history of the time series, but also on its own past variance value. The GARCH(p, q) model can be written as follows

where the necessary condition for stationarity exists

The unconditional variance of the stationary GARCH simulation will be constant and is expressed by the following formula

If the sum of the coefficients is equal to one, then instead of the usual GARCH we get IGARCH, whose unconditional variance is infinite.

Other types of GARCH are planned to be considered in the next practice period.

# 5 Models based on neural networks and decision trees

In the realm of time series forecasting, alongside traditional statistical models, neural networks and decision trees have gained popularity in recent years. Neural networks offer various architectures, but this study focuses solely on three types: Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Convolutional neural networks(CNN). On the other hand, with regards to decision trees, our investigation will encompass XGBoost and CatBoost algorithms.

# 5.1 RNN

RNN is an abbreviation for Recurrent Neural Networks. This type of neural networks implies connections between elements that form a directed sequence.

Training this network is similar to training a conventional neural network using the backpropagation algorithm, but has some differences.

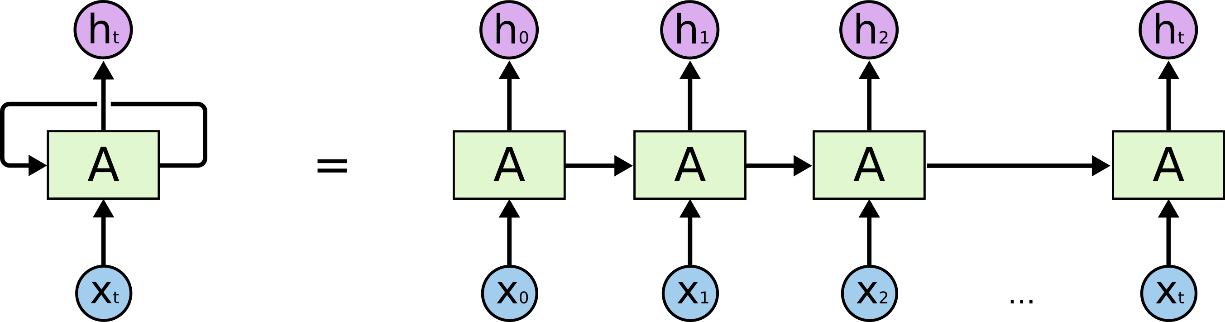


Figure 3. Deployed recurrent neural network[5]

This network, thanks to its loops, allows information to be stored. However, this type of neural networks also has a disadvantage associated with the problem of long-term dependencies [5].

For our task, a variant of the RNN architecture called “many to many” is suitable. Figure 3 shows how this architecture works with data. Red is the input data, blue is the result of the network.

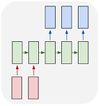


Figure 4. Visualization of many-to-many RNN architecture[6]

This neural network was implemented using the PyTorch library. The implementation has been made public on GitHub[7].

# 5.2 LSTM

LSTMs are long short-term memory networks and are a special kind of RNN capable of learning long-term dependencies. This architecture was specifically invented to avoid the problem of long-term data dependency. Remembering information for a long period of time is practically their default behavior.

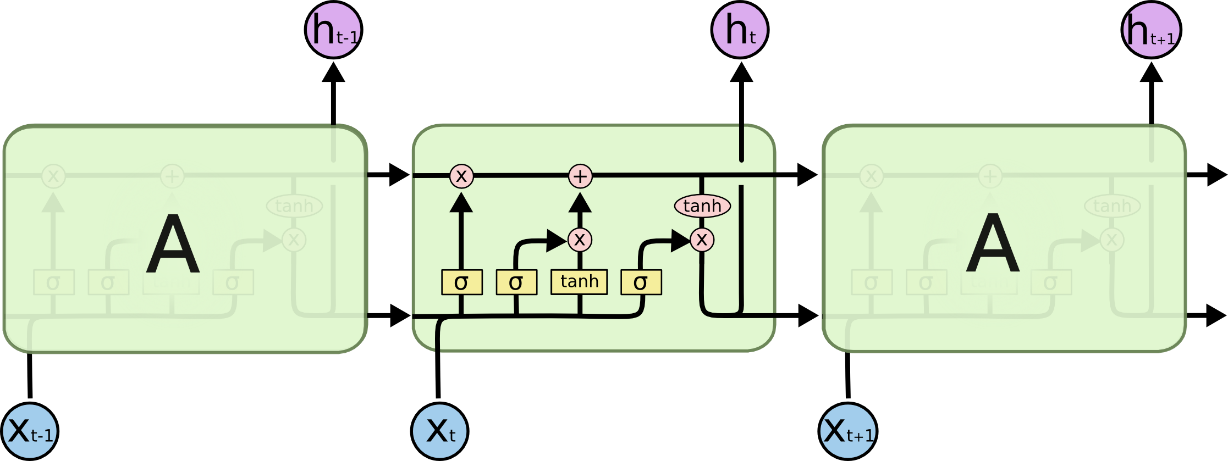


Figure 5. A repeating module in LSTM contains 4 interacting layers[5]. The yellow rectangle is the layers of the neural network, the red ellipse is a pointwise operation, the regular arrow is the transfer vector, the branching arrow is the copy vector, the connecting arrow is the concatenation vector.

The LSTM has the ability to remove and add information to the cell's state, carefully managed by structures called gates.

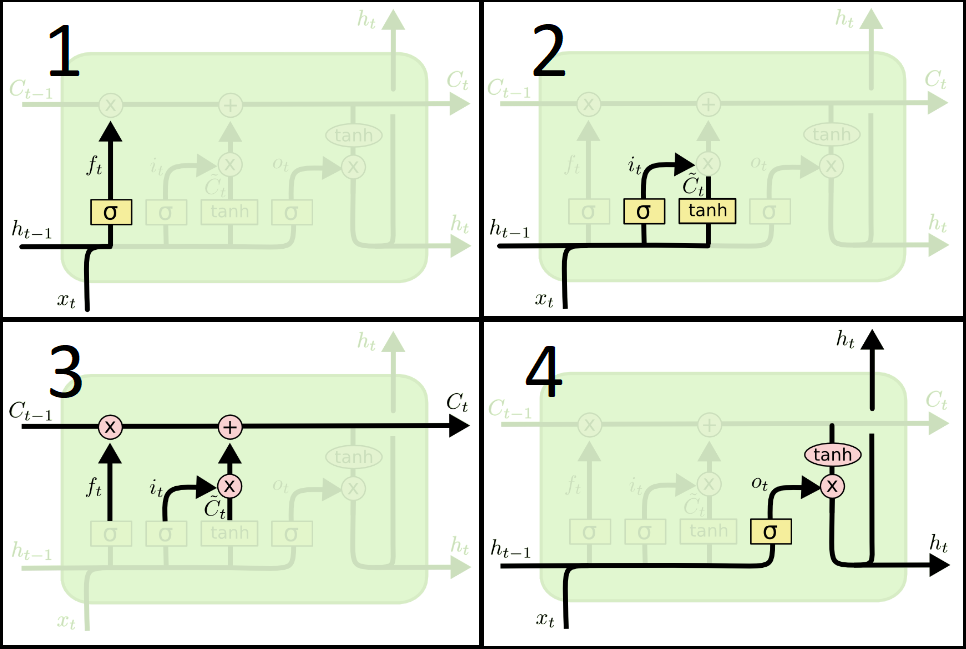


Figure 6. Demonstration of LSTM operation.

The basic LSTM variant is expressed as follows

where the value of each element can be seen in Figure 6. This architecture has several more variations, but for this study it was decided to take the basic version.

LSTM is more commonly used for time series forecasting because this architecture allows patterns in the data to be stored and changed as a result of the further appearance of the data and learning from it, which usually happens with time series. More often than not, the behavior of a time series changes over time.

# 5.3 GRU

One key distinction is that the GRU architecture incorporates a "forget gate" similar to LSTM, which allows the network to selectively retain or discard information from the previous time step. However, GRU has fewer parameters compared to LSTM, making it computationally more efficient.

Unlike LSTM, the GRU architecture does not possess an "output gate." Instead, it utilizes an "update gate" and a "reset gate" to control the flow of information. The update gate determines the extent to which the previous hidden state should be carried forward, while the reset gate controls the degree of influence of the previous hidden state on the current time step.

Additionally, the GRU architecture offers various options for a fully closed block, where the gating mechanism is performed using the previous hidden state and input combinations in different ways. This allows for flexibility in modeling temporal dependencies. There is also a simplified form known as the "minimal gating" block, which further reduces the complexity of the gating mechanism.

These modifications in the GRU architecture enable it to effectively capture dependencies in sequential data while requiring fewer parameters compared to LSTM. The choice of architecture depends on the specific task and the trade-off between model complexity and computational efficiency.

# 5.4 CNN

CNN are a class of neural networks that excel in processing data with spatial properties, such as images. However, they can also be effectively employed for time series analysis.

When applied to time series data, a CNN architecture typically encompasses the following key components:

1. Input Layer: The input layer accepts the time series data, which can be represented as a one-dimensional array or matrix with a single dimension (e.g., time).
2. Convolutional Layer: The convolutional layer is the primary building block of a CNN and enables the extraction of features from the input data. This layer applies filters to the input, accentuating characteristics like periodic fluctuations and trends.
3. Subsampling Layer: Following the convolutional layer, the subsampling layer is introduced to decrease data dimensionality and eliminate noise. This aids in mitigating overfitting and enhancing the model's ability to generalize.
4. Fully Connected Layer: The fully connected layer, a conventional neural network layer, establishes connections between each neuron and all neurons in the preceding layer. This layer combines the features learned in earlier layers to make informed decisions.
5. Output Layer: The output layer predicts future values of the time series.

In a time series CNN architecture, multiple convolutional and subsampling layers can be employed to progressively extract more abstract features from the data. Additionally, recurrent layers such as LSTM and GRU can be incorporated to account for temporal dependencies between time points.

The architecture of a CNN for time series can be deep and intricate, depending on the specific task and the volume of data available. Nevertheless, these models have demonstrated promising performance in time series forecasting and feature extraction.

# 5.5 Decision trees

This forecasting methodology entails a comprehensive review of pertinent literature and the establishment of a framework for validation purposes. In order to further enhance the practical application of this approach, a detailed analysis is currently being conducted, and the outcomes will not be presented at this juncture.

# 6 Conclusion

In this study, work was conducted to search for data, process it, and prepare it for further analysis. Additionally, an analysis of existing architectures for time series forecasting was performed, including statistical methods as well as neural network methods and decision trees.

Based on the results of the study, it is not yet possible to determine the potential of neural networks for time series forecasting, as only an initial implementation of networks with a single layer was performed without exploring different layer combinations and architectures to obtain better results. It is also important to remember that the performance of a neural network depends not only on its architecture but also on proper data preparation and feature engineering.

In the future, the plan is to conduct forecasting using neural networks and decision trees, compare the results of all the methods used, and identify the most successful approach to answer the question of whether neural networks can be effectively applied for time series forecasting.

# References

1. https://www.finam.ru/profile/fyuchersy-usa/nq-100-fut/export/?market=7&em=21719&token=&code=NDX&apply=0&df=1&mf=0&yf=2010&from=01.01.2010&dt=1&mt=0&yt=2014&to=01.01.2014&p=2&f=NDX\_100101\_140101&e=.txt&cn=NDX&dtf=1&tmf=1&MSOR=1&mstime=on&mstimever=1&sep=1&sep2=1&datf=1&at=1

2. https://www.finam.ru/profile/fyuchersy-usa/sandp-fut/export/?market=7&em=108&token=&code=SP&apply=0&df=1&mf=0&yf=2010&from=01.01.2010&dt=1&mt=0&yt=2014&to=01.01.2014&p=2&f=SP\_100101\_140101&e=.txt&cn=SP&dtf=1&tmf=1&MSOR=1&mstime=on&mstimever=1&sep=1&sep2=1&datf=1&at=1

3. Engle, Robert F, Autoregressive Conditional Heteroskedasticity with Esti-mates of the Variance of United Kingdom Inflation. 1982, p. 987–1007, Econometri-ca.

4. Bollerslev, Tim, Generalized Autoregressive Conditional Heteroskedastici-ty, 1986, p. 307–327, Journal of Econometrics.

5. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

6. https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B5\_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5\_%D1%81%D0%B5%D1%82%D0%B8

7. https://github.com/SkivHisink/MasterWork

8. https://web.archive.org/web/20211110112626/http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/

9. https://arxiv.org/abs/1412.3555

10. https://www.sciencedirect.com/science/article/pii/S2212827121003796?ref=pdf\_download&fr=RR-2&rr=79fdf5739bb63aad

11. Kantorovich G.G. Lecture and Methodological Materials

12. Time Series Forecasting in Python, Marco Peixeiro, ISBN: 9781617299889б 2022г.

13. Practical Time Series Analysis, Aileen Nielsen, ISBN: 9781492041658

14. Introduction to Time Series and Forecasting Second Edition, Peter J. Brockwell Rich-ard A. Davis, ISBN 0387953515, 2002 г.

15. Machine Learning for Time Series Forecasting with Python, Francesca Lazzeri, ISBN: 9781119682363, 2021г.