Automatic Time Series Forecasting with GRNN: A Comparison with Other Models

Francisco Martínez[1](✉), Francisco Charte[1], Antonio J. Rivera[1], and María P. Frías[2]

[1] Andalusian Research Institute in Data Science and Computational Intelligence, Computer Science Dept., Universidad de Jaén, Jaén, Spain
{fmartin,fcharte,arivera}@ujaen.es

[2] Statistics and Operations Research Dept., Universidad de Jaén, Jaén, Spain
mpfrias@ujaen.es

Abstract. In this paper a methodology based on general regression neural networks for forecasting time series in an automatic way is presented. The methodology is aimed at achieving an efficient and fast tool so that a large amount of time series can be automatically predicted. In this sense, general regression neural networks present some interesting features, they have a fast single-pass learning and produce deterministic results. The methodology has been implemented in the R environment. A study of packages in R for automatic time series forecasting, including well-known statistical and computational intelligence models such as exponential smoothing, ARIMA or multilayer perceptron, is also done, together with an experimentation on running time and forecast accuracy based on data from the NN3 forecasting competition.

Keywords: Time series forecasting · General regression neural networks · Automatic forecasting

1 Introduction

Automatic time series forecasting allows to predict the future behavior of a time series with a minimal human intervention. This can be very useful when either the user is not an expert on a forecasting methodology or the volume of time series to be forecast is high enough to prevent the use of a human assisted procedure. In this later case, it would also be desirable an efficient forecasting methodology.

In this paper we present an automatic time series forecasting scheme based on generalized regression neural networks [16], which are a variation to radial basis function networks (RBFN). Generalized regression neural networks (GRNNs)

This work is supported by the Spanish National Research Project TIN2015-68454-R.

have a single-pass learning so they can learn quickly. Furthermore, they produce deterministic results, avoiding the need to train several neural networks to achieve accurate and stable predictions.

Our scheme has been implemented in the R environment. We have considered interesting to compare this new method with other automatic methodologies for time series forecasting currently found in R as packages. In this comparison we have used both statistical and computational intelligence based techniques such as exponential smoothing, ARIMA, k-nearest neighbors (KNN) or multilayer perceptron. This comparison can shed some light on the controversial subject of whether statistical methodologies are better than computational intelligence ones [6].

The rest of the paper is structured as follows. In Sect. 2 generalized regression neural networks are analyzed. Section 3 explains our methodology to apply GRNNs to time series forecasting. In Sect. 4 R packages for automatic time series forecasting based on computational intelligence and statistical techniques are described. Furthermore, an R package that can be applied to combine the forecasts of several models is also discussed. In Sect. 5 a comparison among the different analyzed methods is done using data from the NN3 forecasting competition. Finally, Sect. 6 draws some conclusions.

2 Generalized Regression Neural Networks

A general regression neural network is a variant of a RBF network [15] characterized by a fast single-pass learning. A GRNN consists of a hidden layer with RBF neurons. Normally, the hidden layer has so many neurons as training examples. The center of a neuron is its associated training example and so, its output gives a measure of the closeness of the input vector to the training example. Commonly, a neuron will use the multivariate Gaussian function:

$$G(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \tag{1}$$

where x_i and σ are the center and the smoothing parameter respectively—x is the input vector.

Given a training set consisting of n training patterns—vectors $\{x_1, x_2, \ldots x_n\}$—and their associated n targets, normally scalars—$\{y_1, y_2, \ldots y_n\}$—, the GRNN output for an input pattern x is computed in two steps. First, the hidden layer produces a set of weights representing the closeness of x to the training patterns:

$$w_i = \frac{\exp\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{\|x-x_j\|^2}{2\sigma^2}\right)} \tag{2}$$

Note that the weights decay exponentially with distance to the training pattern. The weights sum to one and represent the contribution of every training pattern

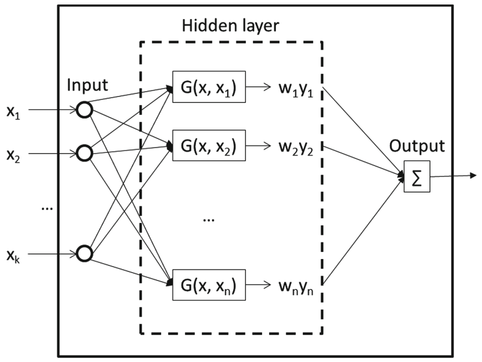


Fig. 1. Topology of a GRNN.

to the final result. The GRNN output layer computes the output as follows:

$$\hat{y} = \sum_{i=1}^{n} w_i y_i \tag{3}$$

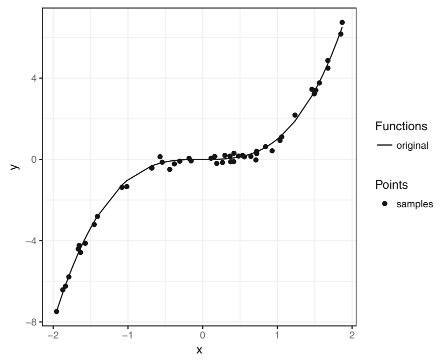
so a weighted average of the training targets is obtained, where the weights decay exponentially with distance to the training patterns—see Fig. 1. The smoothing parameter controls how many targets are important in the weighted average. When σ is very large the result is close to the mean of the training targets because all of them have a similar weight. When σ is small only the closest training targets to the input vector have significant weights. In Fig. 2(a) the function x^3 over the interval $[-2, 2]$ is shown, together with a sample of 50 (x, y)-values of the function with an added random noise. In Fig. 2(b) the values of the function are predicted from the sample of 50 pairs of (x, y)-values with noise using GRNN regression and a small smoothing parameter. In Fig. 3(a) σ is larger and the regression seems better. Finally, in Fig. 3(b) σ seems too high. As can be seen in the example, GRNN regression is very sensitive to the smoothing parameter.

3 Time Series Forecasting with GRNN

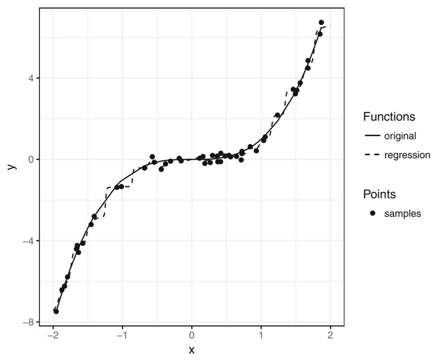
In this section the methodology that has been developed to forecast a time series with GRNN is explained. The goal is to obtain a fast and automatic tool. In the following subsections the different design choices are explained.

3.1 Preprocessing

In our methodology the time series has been scaled to the range $[0, 1]$. However, no other preprocessing, including detendring or deseasonalizing—that is, transforming the time series to remove trend or seasonality—, is done.

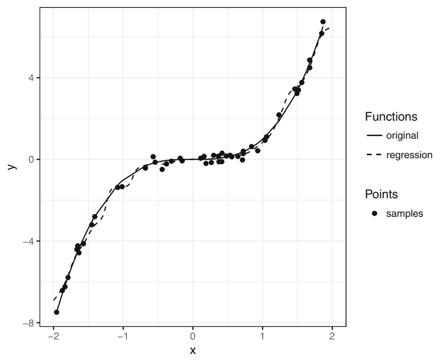


(a) Function x^3 and a sample with random noise

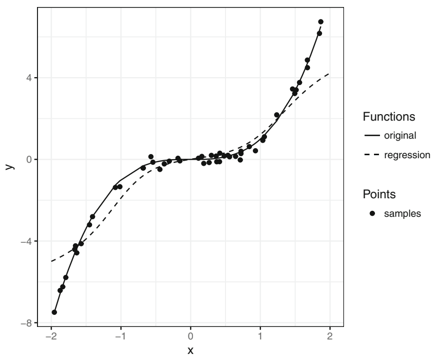


(b) GRNN with $\sigma = 0.05$

Fig. 2. A sample of a function with random noise and its regression with GRNN



(a) $\sigma = 0.1$



(b) $\sigma = 0.5$

Fig. 3. Regression with GRNN and different smoothing parameters

3.2 Autoregressive Lags and Number of Neurons

In order to select the autoregressive lags the following strategy will be used. If the time series is seasonal and the length of the seasonal period is m, then m consecutive lags, starting from lag 1, are used. For example, for quarterly data lags 1:4 are used and for monthly data lags 1:12. This way, seasonal patterns can be captured more easily. Let us see why. In Fig. 4 an artificial quarterly time series with a strong seasonal pattern is shown. The first quarter has a mean level higher than the other quarters that have a similar level. The autoregressive lags are lags 1:2 and we want to generate a one-step ahead forecast. These autoregressive lags can lead to an unsuitable forecast. As can be seen in the figure, the closest training pattern to the input pattern has a fourth quarter as target, when a first quarter value is going to be predicted. In Fig. 5 lags 1:4 are used and the target associated to the closest training pattern is a first quarter value.

If the time series is not seasonal, for example yearly data, then the lags with significant autocorrelation in the partial autocorrelation function (PACF) are selected. Although the PACF only tests for linear relationship, experience has shown us that this is an effective way of selecting input variables [14]. If none of the previous two conditions are met, then lags 1:5 are used. Note that this way of selecting the autoregressive lags is quite fast.

The GRNN will use so many hidden neurons as training examples. When the number of training examples is very high GRNN can use clustering to reduce the number of hidden neurons, but in our case this technique will not be used.

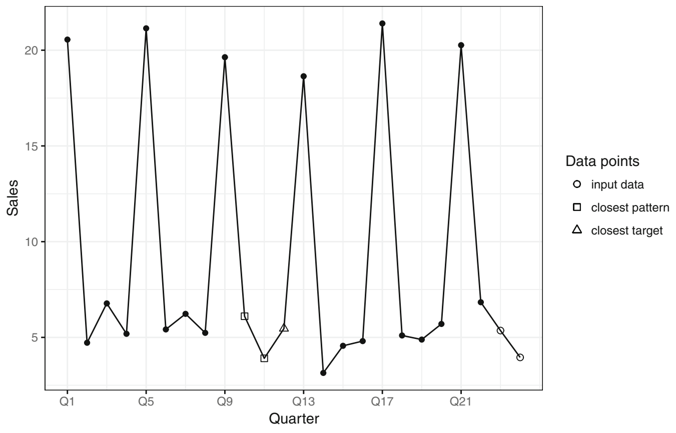


Fig. 4. Quarterly time series with a strong seasonal pattern and lags 1:2.

3.3 Selecting the Smoothing Parameter

As previously mentioned, GRNN is very sensitive to the smoothing parameter so it is vital to select a suitable value for it. In order to make a good choice we have applied an optimization tool for finding σ using the rolling origin technique. The historical data is divided into a training and a validation set and σ is selected so that it minimizes a forecast accuracy measure for the validation data using the training data.

3.4 Multi-step Ahead Strategy

When more than one future value of a time series has to be predicted a multi-step ahead strategy must be applied [2]. The classical strategies are direct, Multiple-Input Multiple-Output (MIMO) and iterative. We rule out MIMO because normally produces less accurate forecasts. There is no clear evidence that the direct strategy outperforms the iterative strategy so we choose the last one because it is straightforward and faster.

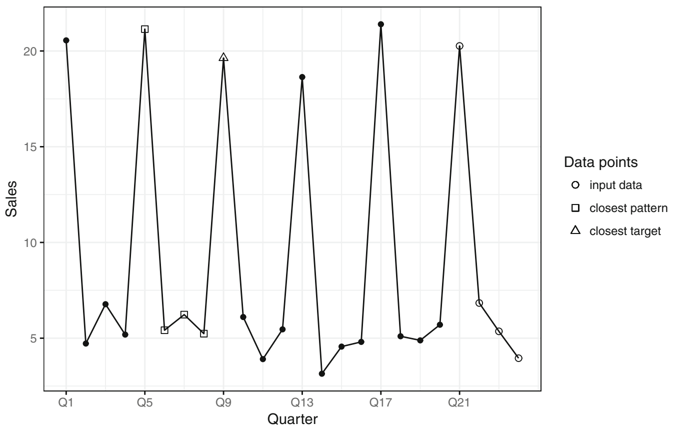


Fig. 5. Quarterly time series with a strong seasonal pattern and lags 1:4.

4 Automatic Time Series Forecasting in R

In this section we briefly describe other automatic time series forecasting methods that can be found in the R environment. We have classified them according to whether they are based on computational intelligence or statistical techniques. In the next subsections they are described. For every method, it is also explained how it works when it is used with default parameters. In general, when you call a forecasting function with default parameters the function tries to automatically select the best model to predict the time series.

4.1 Computational Intelligence Methods

This section analyzes the computational intelligence methods related to time series forecasting. We first have searched in the CRAN task view Time Series Analysis [5] to see what R packages are related to time series forecasting. Although CRAN—the Comprehensive R Archive Network—contains a lot of packages for regression based on computational intelligence techniques [3], just a few are specially devoted to time series forecasting.

The *tsfknn* Package. This package allows to forecast a time series using KNN regression, the methodology used is partially derived from this study [14]. When forecasts are generated with default parameters the package employs the following strategies:

- No preprocessing is done for detendring, deseasonalizing or scaling the time series.
- Instead of using a unique k parameter, three preset k parameters are used. Three diverse KNN models are built, each one with a different value of the preset k values. Forecasts are generated using the three KNN models and are

averaged to obtain the final forecast. The goal of this strategy of employing several preset k values is to avoid the use of a slow optimization technique to find the k parameter that best fit the historical observations.
- The autoregressive lags are selected in the same way as described in the proposed GRNN methodology.
- For multi-step ahead forecasts the recursive or iterative strategy is used.

The *nnetar* Function from the Forecast Package. The *nnetar* function allows to forecast a time series by means of a multilayer perceptron—MLP. This function is part of the outstanding *forecast* package [8], which is described in an online book [11]. When this function is invoked with default parameters it works as follows:

- As preprocessing, the time series is scaled by subtracting the mean and dividing by the standard deviation.
- The autoregressive lags are selected as follows. For non-seasonal time series, p consecutive lags are selected, starting from lag 1. The number of lags is the optimal number of lags of an autoregressive ARIMA model—i.e., an ARIMA model with 0 differences and no moving average terms—using the AIC to select the model. For seasonal time series, the p consecutive lags are chosen from the optimal AR linear model fitted to the seasonally adjusted data. Furthermore, lag m, where m is the seasonal period, is also used. For example, for monthly data lag 12 is also used.
- Only a hidden layer is used and its number of neurons is half of the number of input nodes—autoregressive lags—plus 1.
- In order to achieve more stable and accurate forecasts, 20 neural networks are fitted with different random starting weights. To produce the final forecast, the forecasts of these 20 networks are averaged.
- For multi-step ahead forecasts the recursive or iterative strategy is used.

The *mlp* Function from the nnfor Package. The *mlp* function from the *nnfor* package also implements multilayer perceptron for time series forecasting. Let us see how it automatically predicts a time series:

- As preprocessing, the time series is linearly scaled to $[-.8, .8]$. A test for finding trend is done and first differences are applied if necessary. Then the series is tested for seasonality, if the test successes seasonal differences are taken.
- The autoregressive lags are selected as described here [7]. The model can also include seasonal dummy variables.
- By default only a hidden layer with 5 nodes is used. Setting the appropriate parameters the number of neurons can be selected using an optimization algorithm.
- 20 neural networks are fitted with different random starting weights. The forecasts of these 20 networks are combined using the median operator to obtain the final forecast.
- For multi-step ahead forecasts the recursive or iterative strategy is used.

4.2 Statistical Models

In this section we analyze the two workhorses of statistical models for time series forecasting: exponential smoothing [10] and ARIMA [4]. The previously mentioned *forecast* package includes implementations of both methods with automatic selection of models and model parameters.

Exponential Smoothing. Exponential smoothing encompasses a set of models—e.g., SES, Holt or Holt-Winters—to predict a time series taking into account its trend and seasonal components. The function *ets* uses the AICc criterion to select an appropriate exponential smoothing model—for example, a model with damped trend and multiplicative seasonality—among all the possible exponential smoothing models.

ARIMA. Normally, an ARIMA model is selected by an expert after making the series stationary, analyzing the autocorrelation and partial autocorrelation functions and trying several models. The *auto.arima* function of the *forecast* package uses a variation of the Hyndman-Khandakar algorithm [8] to select an ARIMA model. Basically, it uses unit root tests to check stationarity and possibly take differences. Several ARIMA models are fitted and the AICc criterion is used to select the best one. The search for the ARIMA model is not exhaustive, so the selected model might not be the optimal one—according to AICc—but a good model is found.

4.3 A Combination of Methods

Since Bates and Granger [1] proposed to combine the forecasts of several methods, forecasting by combining different techniques has not stopped growing. As an example, the recent M4 forecasting competition [13], wherein 100,000 time series had to be forecast, has been dominated by combinations of models.

Taking into account the success of the ensembles of methods we have considered interesting to include a combination of models in our comparison. The *forecastHybrid* package combines, by default, seven models from the *forecast* package averaging their forecasts. Three of the models used in the combination have been described here: *nnetar*, *auto.arima* and *ets*—exponential smoothing. The package can be applied in an automatic way in which the user only sets the time series and the forecast horizon.

5 Experimentation

In this section a comparison among the proposed time series forecasting methodology based on GRNN and the other techniques explained in the previous section is carried out. To compare the methods the data from the NN3 competition [6] has been used. In this competition 111 time series drawn from the M3 monthly industry data [12] were used. The data set contains a balanced mix of 25 short seasonal series, 25 short non-seasonal series, 25 long seasonal series, 25 long non-seasonal series and a collection of 11 experimental series. Short series have about 52 observations per series and long series more than 120 observations.

In the NN3 competition the next 18 future months for all the time series should be predicted. To assess the accuracy of a forecast the NN3 organizers decided to use the sMAPE—symmetric absolute percentage error. Given the forecast F for a NN3 time series with actual values X:

$$\text{sMAPE} = \frac{1}{18}\sum_{t=1}^{18}\frac{|X_t - F_t|}{(|X_t| + |F_t|)/2}100$$

The sMAPE of each series will then be averaged over all the 111 time series for a global mean sMAPE. Although some experts discourage the use of sMAPE [9], it was the main measure for assessing forecast accuracy in the NN3 competition, so we decided to use it to compare our results with the NN3 contenders.

In Table 1 the results of our experiments are shown. The methods are listed in the first column, sorted by forecast accuracy according to the global mean sMAPE—second column—on the 111 time series, computed as described previously. The third column shows the average rank of every method over the 111 time series. The next four columns include the average sMAPE of the methods on the different data conditions evaluated in the NN3 competition: S.S. (short seasonal), S.N. (short non-seasonal), L.S. (long seasonal) and L.N. (long non-seasonal). The last column of the table shows the time in minutes needed by the method to forecast the 111 time series. For benchmarking purposes, the sMAPE of the 8 top contenders of the NN3 competition is shown in Table 2. In the ID field of this table, B stands for statistical benchmark and C for computational intelligence method.

After these results several conclusions can be drawn:

- The combination method is the winner, being its accuracy similar to the winner of the NN3 competition. This is an outstanding result, because the methods we have compared are applied automatically, in the sense that no study of the characteristics of the time series has been taken into account in order to apply the methods. The contenders of the NN3 competition knew the historical data and they could analyze their features to improve their models. The fact that the winner is a combination methodology is consistent with previous studies [12]. Its performance is very robust, beating the other methods over all data conditions.
- The statistical methods, exponential smoothing and ARIMA, have obtained a similar global sMAPE. Their results clearly outweigh the computational intelligence techniques. This superiority also corroborates other comparisons [13]. Furthermore, the *ets* function is remarkably fast. In spite of having a better mean rank, *auto.arima* has a worse global sMAPE due to its poor performance in the short seasonal series.

– The GRNN methodology developed in this paper has achieved slightly better performance than the KNN and *nnetar*. No wonder that KNN and GRNN get similar results because both are based on combining the targets of the patterns that are similar to the input pattern. GRNN is a bit slow because the smoothing parameter is selected by optimization.
– The *nnetar* function, based on multilayer perceptron, has achieved modest results, being beaten by a simple method such as a KNN. However, its mean rank is lower than GRNN or KNN and its results on long series are acceptable. Maybe, a proper training is not possible with short series.
– The result of the *mlp* function from the *nnfor* package is a bit disappointing. It is nearly the worst method over all the conditions. Its default configuration can possibly be improved.

Table 1. Comparison of the different methods.

Method	sMAPE	Rank	S.S.	S.N.	L.S.	L.N.	Time
forecastHybrid	14.85	2.79	13.66	19.21	9.38	16.92	16:11
ets	15.52	3.67	14.69	19.57	10.50	17.53	01:29
auto.arima	15.64	3.47	16.66	19.36	10.14	17.01	09:12
GRNN	16.71	4.29	16.11	19.57	13.80	18.35	06:45
tsfknn	16.96	4.31	15.38	21.90	12.85	19.18	00:03
nnetar	17.05	4.18	16.00	24.73	11.94	18.26	00:14
mlp	21.22	5.30	22.28	24.31	16.09	20.23	11:10

Table 2. The top NN3 competition contenders.

ID	Method	sMAPE
B09	Wildi	14.84
B07	Theta	14.89
C27	Echo state networks	15.18
B03	ForecastPro	15.44
B16	DES	15.90
B17	Comb S-H-D	15.93
B05	Autobox	15.95
C03	Linear model + GA	16.31

Figure 6 shows the boxplots of the accuracy according to sMAPE for the different methods over the 111 time series. We would have liked to add some recurrent network to the comparison, but to our knowledge R lacks of a package of that kind for time series forecasting.

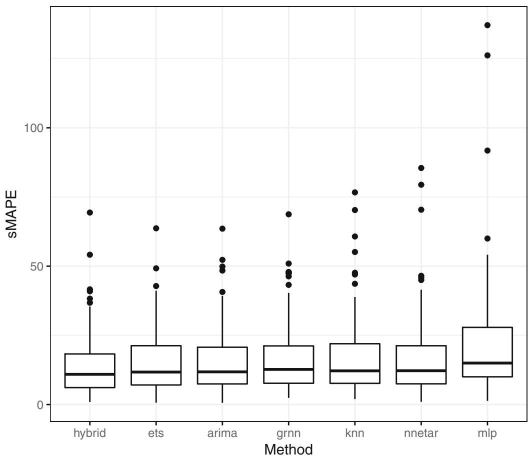


Fig. 6. Boxplots of the different methods.

6 Conclusions

In this paper a methodology based on GRNN regression for forecasting time series in an automatic way has been presented. The methodology uses straightforward strategies in order to obtain a fast forecasting tool. This goal has been facilitated by the intrinsic features of GRNN regression, such a single-pass learning or deterministic predictions. Currently, the bottleneck of the tool is the selection of the smoothing parameter. Packages for automatic time series forecasting in the R environment have also been described, together with a comparison among the studied tools in terms of forecasting accuracy and running time. In this comparison, based on monthly time series, statistical models seem to be more accurate and the proposed methodology has achieved good result among the computational intelligence models.

References

1. Bates, J.M., Granger, C.W.J.: The combination of forecasts. Oper. Res. Q. **20**, 451–468 (1969)
2. Ben Taieb, S., Bontempi, G., Atiya, A.F., Sorjamaa, A.: A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Syst. Appl. **39**(8), 7067–7083 (2012)
3. Bhatia, A., Chiu, Y.: Machine Learning with R Cookbook: Analyze Data and Build Predictive Models. Packt Publishing, Birmingham (2017)
4. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, 4th edn. Wiley, Hoboken (2008)
5. CRAN Task View: Time Series Analysis. https://cran.r-project.org/view=TimeSeries. Accessed 26 Feb 2019

6. Crone, S.F., Hibon, M., Nikolopoulos, K.: Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. Int. J. Forecast. **27**(3), 635–660 (2011)
7. Crone, S.F., Kourentzes, N.: Feature selection for time series prediction - a combined filter and wrapper approach for neural networks. Neurocomputing **73**(10), 1923–1936 (2010)
8. Hyndman, R., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. J. Stat. Softw. **27**(1), 1–22 (2008)
9. Hyndman, R.J., Koehler, A.B.: Another look at measures of forecast accuracy. Int. J. Forecast. **22**(4), 679–688 (2006)
10. Hyndman, R.J., Koehler, A.B., Ord, J.K., Snyder, R.D.: Forecasting with Exponential Smoothing: The State Space Approach. SSS. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71918-2
11. Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice, 2nd edn. OTexts, Melbourne (2018). OTexts.com/fpp2. Accessed 26 Feb 2019
12. Makridakis, S., Hibon, M.: The M3-competition: results, conclusions and implications. Int. J. Forecast. **16**(4), 451–476 (2000)
13. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The M4 competition: results, findings, conclusion and way forward. Int. J. Forecast. **34**(4), 802–808 (2018)
14. Martínez, F., Frías, M.P., Pérez, M.D., Rivera, A.J.: A methodology for applying k-nearest neighbor to time series forecasting. Artif. Intell. Rev. (2017)
15. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Comput. **1**(2), 281–294 (1989)
16. Specht, D.F.: A general regression neural network. Trans. Neural Netw. **2**(6), 568–576 (1991)