UDC 004.032.26

COMPARATIVE ANALYSIS OF MODELS AND FORECASTING METHODS

D. VASILYEVA, D. GLUKHOV
Polotsk State University, Belarus

In this article, work has been done on the study and comparison of forecasting methods, namely the methods: ARIMA, LSTM, Facebook Prophet. Examples of the work of these methods are given, and the best one for the implementation of the "Forecast of natural gas consumption in Belarus" is selected.

Study of forecasting methods. Forecasting gas consumption is the basis not only for planning purchases of natural gas from gas traders, but, no less important, for developing measures to manage energy consumption in the municipal economy, especially during the heating season.

One of the best methods to construct a forecast of the consumption of a particular resource is a neural network approach, since it is free from model constraints and is equally suitable for linear and complex nonlinear problems, as well as classification problems. Technically, training consists in finding the coefficients of connections between neurons. In the process of training, the neural network is able to identify complex dependencies between input data and output data, as well as perform generalization. This means that in case of successful training, the network will be able to return the correct result based on data that was absent in the training sample, as well as incomplete and / or "noisy", partially distorted data.

Therefore, there are many different approaches for time series forecasting such as ARIMA, LSTM, Facebook Prophet, regression models, etc.

A popular and widely used statistical method for forecasting time series is the ARIMA model. It explicitly serves a set of standard time series data structures and, as such, provides a simple yet powerful technique for making sophisticated time series predictions.

ARIMA stands for AutoRegressive Integrated Moving Average. It is a model class that captures a set of different standard time structures in time series data.

Using ARIMA it is possible to make predictions on non-stationary data due to the introduction of integration into the model. This is achieved by taking differences - subtracting the levels of the time series from each other.

Given the seasonality of the time series, short-term components are likely to make a significant contribution to the model. Thus, the model also needs to take into account seasonality - seasonal ARIMA. The most important steps are the estimation of model coefficients. If variance grows over time, variance-stabilizing transformations and taking differences should be used.

One of the obvious shortcomings of the models is the requirement for data series: to build an adequate ARIMA model, at least 40 observations are required, and for SARIMA, about 6–10 seasons, which is not always possible in practice.

The second serious drawback is the inadaptiveness of autoregressive models: when new data is received, the model must be periodically reevaluated, and sometimes re-identified.

The third disadvantage is that building a satisfactory ARIMA model is resource and time consuming. The very construction of the model is more likely an "art", i.e. requires a lot of experience on the part of the forecaster.

The whole construction of ARIMA models is based on the assumption that the time series is generated infinitely in accordance with some function, the parameters of which we need to identify and estimate, i.e. the ARIMA approach is based on the assumption of the frozen nature of the ongoing processes, evolutionary nature as such is not taken into account in the model. This is primarily due to the fact that the models were originally developed for modeling physical and technical processes, in which almost all types of processes are described either as stationary or as stationary in differences. From which we can conclude that the use of these models will not allow us to give accurate predictions.

To solve the problem of storing a small number of previous observations, LSTM networks have been developed.

Long short-term memory neural network is an artificial convolutional neural network used in the field of deep learning. Unlike conventional feedforward neural networks, LSTM networks have feedback loops. Such networks are capable of processing not only individual single data, but also entire data sequences. LSTM neural networks are well suited for classification, processing and forecasting based on time series, where interrelated phenomena can occur with an indefinite time lag. This time lag leads to difficulties in using classical neural networks in solving these problems due to the fading of the gradient, while LSTM networks are insensitive to the value of the time lag.

There are 3 types of layers in LSTM networks:

1. Forget gate layer
2. Layer "memory" (Memory gate)
3. Output layer (Output gate)

First, the "forgetting filter layer" determines what information can be forgotten or left behind. The values of the previous output and current input are passed through the sigmoidal layer. The obtained values are in the range [0; one]. Values closer to 0 will be forgotten and values closer to 1 will be left.$h_{t-1}x_t$

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \quad (1)$$

Next, it is decided what new information will be stored in the cell state. First, a sigmoidal layer called the "input filter layer" determines which values to update. The tanh layer then builds a vector of new candidate values c that can be added to the cell state.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$C_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$

To replace the old state of the cell with the new state, it is necessary to multiply the old state by forgetting what you decided to forget earlier. Then we add $C_{t-1}C_t f_t i_t *$... These are the new candidate values multiplied by $C_t t$ - how much to update each of the state values.

At the last stage, it is determined what information will be obtained at the output. The output will be based on our cell state, with some filters applied to it. First, the values of the previous output and the current input are passed through the sigmoidal layer, which decides what information will be output from the cell state. The cell state values are then passed through the tanh layer to output values in the -1 to 1 range and are multiplied with the output values of the sigmoidal layer, which allows only the information required to be displayed.h_{t-1}

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (4)$$

$$h_t = o_t * tanh(C_t) \quad (5)$$

where is the input vector; - output vector, - vector of states;x_t $h_t C_t$ *W* and *b*- matrices of parameters and a vector; is the vector of the forgetting gate; the weight of storing old information; is the vector of the input gate; the weight of receiving new information; is the vector of the output gate, the candidate for the output. f_t $i_t o_t$ received in this way and are transmitted further along the chain.$h_t C_t$

Facebook Prophet is an open source forecasting tool available in Python, and R. Prophet is optimized for business forecasting tasks, which typically have any of the following characteristics:

- hourly, daily or weekly observations with a history of at least several months (preferably a year);
- strong multiple seasonality: day of the week and season;
- a reasonable number of missing observations or large outliers;
- historical changes in trends;
- trends, which are non-linear growth curves when the trend reaches a natural limit or reaches saturation.

There are anti-aliasing options for seasonality that allowing to tune how closely historical cycles are matched, and there are anti-aliasing options for trends that allowing to tune how aggressively changes are followed in historical trends. For growth curves, you can manually specify "powers" or the upper limit of the growth curve, allowing you to enter your own preliminary information about how the forecast will rise (or decline).

In fact, Prophet is an additive regression model consisting of the following components:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \quad (6)$$

where the trend is; - seasonal components; - abnormal days; - errors.$g(t)$ $s(t)$ $h(t)$ ϵ_t

1. Seasonal components are responsible for simulating periodic changes associated with weekly and yearly seasonality. Weekly seasonality is modeled using dummy variables. 6 additional signs are added, for example, [monday, tuesday, wednesday, thursday, friday, saturday], which take on the values 0 and 1 depending on the date. The sunday attribute corresponding to the seventh day of the week is not added, because it will linearly depend on other days of the week and this will affect the model. The annual seasonality is modeled by Fourier series.$s(t)$

2. A trend is a piecewise linear or logistic function. With a linear function, everything is clear. The logistic function of the form (11) makes it possible to simulate growth with saturation, when with an increase in the indicator, its growth rate decreases. A typical example is the growth of the audience of an application or site. Among other things, the library is able to select the optimal points of trend change based on historical data. But they can also be set manually (for example, if the release dates of new functionality are known, which greatly influenced the key indicators).$g(t)$

3. The component is responsible for user-defined abnormal days, including irregular ones, such as, for example, Black Fridays. $h(t)$

4. The error contains information that is not considered by the model. ϵ_t

$$g(t) = \frac{C}{1 + \exp\left(-k(t-b)\right)} \tag{7}$$

An important idea in Prophet is that by doing more flexible work in adjusting the trend component, we model seasonality more accurately, and as a result, we get a more accurate forecast. The Facebook Prophet developers prefer to use a very flexible regression model instead of the traditional time series model for this task because it gives more modeling flexibility, makes model fitting easier, and handles missing data or outliers more gracefully.

By default, Prophet provides uncertainty intervals for the trend component, simulating future trend changes for a time series. Prophet and its core are implemented in the Stan probabilistic programming language. Stan performs MAP optimization for parameters very quickly (<1 second), which makes it possible to estimate parameter uncertainty using the Hamiltonian Monte Carlo algorithm and allows reuse of the fitting procedure for several interface languages.

Results, discussion and prospects. We have studied the trends of 220 natural gas consumers in the Republic of Belarus over the past 8 years (2012-2020). Moreover, the data of 2012. appear from May and data for 2020. incomplete.

All tracks have the following features:

– The data shows the presence of telemetry errors, emissions and zeros associated with temporary failure of flow sensors, or pressure and temperature, when calculating the flow rate by calculation;

– There are data gaps for the periods of scheduled and emergency gas pipeline repairs;

– There is no exact alignment of timestamps;

– The data does not reflect the fact of gas consumption, but the operating mode of the gas transmission network for the delivery of contract gas to the consumer, and accordingly reflects non-stationary processes (gas injection into the system, gas disassembly from the system in the cylinder mode, when gas flows through the gas distribution station or compressor station does not occur, and consumption gas continues);

– Gas flow rates recorded by sensors lack information on gas density;

– A part of the flow rates was obtained by calculation using a stationary non-isothermal model under boundary conditions specified through pressure and temperature;

Figure 1 shows a graph of forecasting using the ARIMA model, the graph shows the available data and the forecast itself. Calculating the root mean square error of the model, or more simply, the RMSE gave the result 0.1364, from which we can conclude that, although the model has errors, the result is still close to the real values.

Figure 1. - Forecast for 12 weeks according to the ARIMA model (1, 0, 2) (1, 0, 1)

Figure 2 depicts the prediction of the LSTM model. Since this model is designed for forecasting small time series, it starts forecasting from the beginning of the data, gradually improving the forecast accuracy, which is very useful when some data is missing. The graph shows available data, current forecast and future. RMSE for this model = 0.3485, and it performed worse than ARIMA.

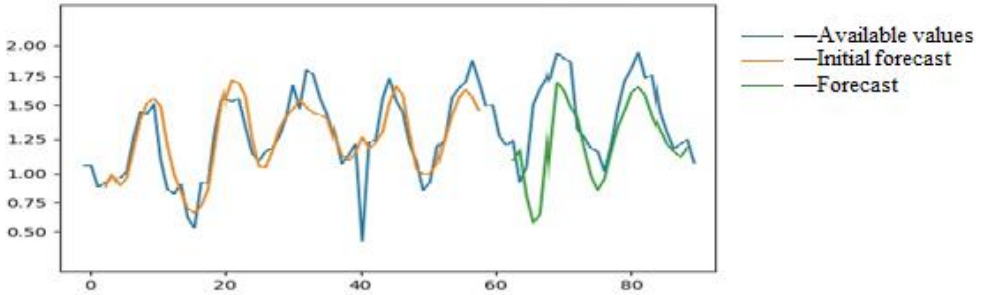

Figure 2. - Forecast for 12 weeks according to the LSTM model

Figure 3 shows Prophet at work. This model always uses only two values: date and numeric. It is designed to work with large amounts of data. It shows the available data, the maximum and minimum possible data and the forecast itself, which, like the LSTM, starts with the beginning of the data. RMSE for this model = 0.2051, but it is worth noting that with each subsequent step the forecast becomes better, based on this, we conclude that more data is needed for a more accurate forecast of this model.

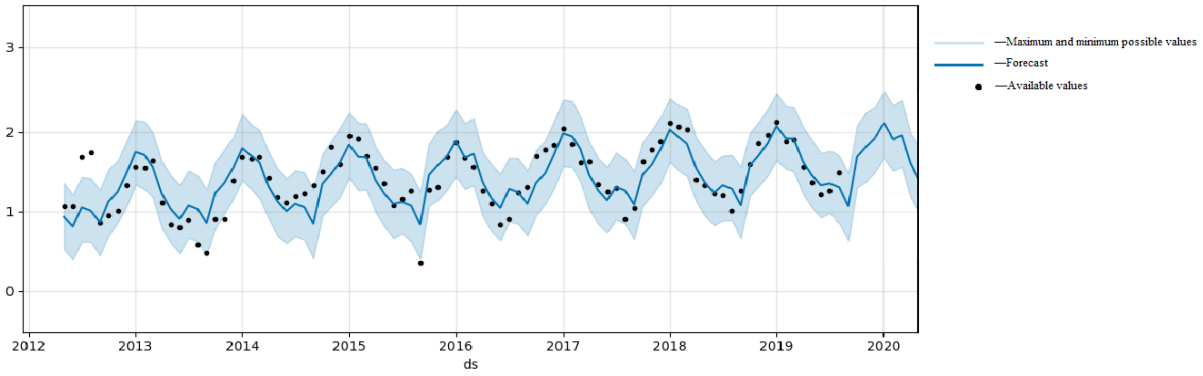

Figure 3. - Forecast for 12 weeks according to the Prophet model

Conclusion. The ARIMA model is much more expensive than the Prophet. For it, it is necessary to investigate the initial series, bring it to a stationary one, select initial approximations and spend a lot of time on the selection of hyper-parameters of the algorithm, but in this case the efforts are not in vain because the ARIMA prediction turned out to be more accurate (RMSEARIM <RMSEProphet). The forecast of the LSTM model has a large error, but perhaps with its help, or with the help of Prophet, we could predict missing values in the past, in order to possibly improve the forecast of other models.

REFERENCES

1. Section "Interval forecasting of time series using recurrent neural networks with long short-term memory" on the Habr website [Electronic resource]. - Access mode: https://habr.com/ru/post/505338/. - Date of access: 11/16/2020.
2. Section "Facebook Prophet" on the site Facebook Research [Electronic resource]. - Access mode: https://research.fb.com/prophet-forecasting-at-scale/. - Date of access: 01.03.2021.
3. Section "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras" on the Machine Learning Mastery website [Electronic resource]. - Access mode: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/. - Date of access: 03/07/2021.
4. Section "Time series analysis using python" on the website habr.com [Electronic resource]. - Access mode: https://habr.com/ru/post/207160/. - Date of access: 03/14/2021.