BJ O&PM
ISSN 2237-8960

Brazilian Journal of Operations & Production Management

ABEPRO
ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA DE PRODUÇÃO

FORECAST OF MULTIVARIATE TIME SERIES SAMPLED FROM INDUSTRIAL MACHINERY SENSORS

Heron Felipe Rosas dos Santos
heron.frs@gmail.com
Fluminense Federal University, Rio das Ostras, RJ, Brazil.

Leila Weitzel Coelho da Silva
leila_weitzel@id.uff.br
Fluminense Federal University, Rio das Ostras, RJ, Brazil.

Ana Paula Barbosa Sobral
ana_sobral@vm.uff.br
Fluminense Federal University, Rio das Ostras, RJ, Brazil.

ABSTRACT

Goal: To evaluate the performance of a set of forecasting methods in the prediction of future values on a dataset of time series collected from sensors installed in an industrial gas turbine.

Design / Methodology / Approach: Forecasting methods tested include the use of multivariate and univariate neural networks (FNN and LSTM), exponential smoothing and ARIMA models.

Results: Results show that the use of ARIMA models to forecast on the dataset is the best default method to apply, and is the only forecasting method that consistently beats a simple naïve no-change model.

Limitation of the investigation: There was a focus on evaluating neural networks. This limited resources available to evaluate other forecasting methods. There is no guarantee that it would not be possible to find neural networks capable of yielding better forecasts than the ones achieved by the best performing methods in this research.

Practical implications: The broadest possible implications of the results are that the best default method to forecast industrial machinery time series is the use of ARIMA models. Additionally, neural networks are not capable of beating methods well stablished within the forecasting community, namely ARIMA models.

Originality / Value: To the best of the authors' knowledge, there is a scarce amount of published evaluations of multiple forecasting methods on data from real machines. This knowledge is useful for the understanding of the best forecasting methods available for the estimation of machine's RUL using sensor time series.

Keywords: Prognostics; Time Series; Forecasting; Neural Networks; ARIMA.

ABEPRO
DOI: 10.14488/BJOPM.2020.010

INTRODUCTION

Condition Based Maintenance (CBM) is a maintenance policy that aims to take maintenance action before a failure happens. CBM times the maintenance action by assessing product condition, and predicting failure based on data gathered from the product. While the technologies and technical methods for CBM are still in their infancy, advancements in information technology have accelerated growth in CBM technology by enabling network bandwidth, data collection and retrieval, data analysis, and decision support capabilities for large datasets of time series. Process data, collected in the form of time series, is often compressed and archived for record keeping and only retrieved for emergency analysis after a fault has occurred. This data could be of tremendous advantage when combined with effective analytics and superior computing power capable of generating knowledge from the data (Qin, 2014; Shin and Jun, 2015). The topic of leveraging embedded sensors, industrial networks and data mining techniques in order to attend high complexity industrial demands also relates to the current research trend on the Internet of Things (IoT). The four basic aspects of IoTs are reliable and accurate data collection; capacity to collect a huge quantity of data; rapid data transmission and automated processes (Lopes Miranda Junior et al., 2017; Alarcón et al., 2016).

Diagnostics and prognostics are two parts of CBM. Diagnostics is a reactive process. It takes place after a fault has already occurred and aims to determine the root cause of the failure. It cannot prevent machine downtime and the corresponding expenses. Prognostics is a proactive process. It assesses and predicts future machine health, which includes detecting incipient failures and predicting remaining useful life (Lee et al., 2014).

There are three classes to the current approaches to prognostics: model based, data driven and hybrid. Model based approaches presume that it is possible to build a mathematical model from the understanding of the physical mechanisms involved in the failure modes of the machine that bases the model. While these approaches have the advantage of providing the ability to incorporate physical understanding of the system, if the understanding of the system degradation is poor, it may be difficult to model the system behavior. Data driven approaches use data gathered from sensors or by the machine operators to track features that indicate the degradation of the system. Data driven approaches can leverage computer intelligence techniques like neural networks and decision trees, or statistical techniques like auto-regressive models (Dragomir et al., 2009).

Historically, for time series in diverse domains, empirical evaluations showed that statistically sophisticated or complex methods do not necessarily produce more accurate forecasts than simpler ones. However, recent evaluations have concluded that complex methods based on computational intelligence and neural networks have caught up, and that simple methods can no longer claim to outperform computer intelligence methods without a proper empirical evaluation (Crone et al., 2011).

Several studies into prognostics have treated it from a time series forecasting perspective. Pham et al. (2012) used an Auto Regressive Moving Average (ARMA) model on baseline data. Pham et al. used deviations from the ARMA model on future values as a degradation index. After the degradation index reaches a threshold, Pham et al. used Cox's PHM (Proportional Hazards Model) to create a survival probability curve as a function of time followed by Support Vector Regression (SVR) to predict remaining useful life. Heng et al. (2009) used an artificial neural network with the most recent values of a condition index (bandpass vibration) as inputs to predict probability of failure in fixed time intervals ahead of the last condition index measure. Heng et al. (2009) benchmarked the proposed model against an Elman Recurrent Neural Network (RNN). Datong et al. (2011) developed a SVR based strategy for on-line prediction of industrial sensor data. The authors tested the strategy on a benchmark dataset. Datong et al. compare their proposed strategy against a different SVR based strategy. Niu and Yang (2010) used a neural network to fuse a set of features into a single value used for condition monitoring. After the condition index reaches a threshold value two non-linear techniques, Dempster-Shafer regression and least-squares support vector machines, predict the future behavior of the monitored index. A weighted average combines the predictions from both methods. Cho et al. (2016) developed a hybrid approach to predict the next failure time of a centrifugal compressor using vibration data. Bellow a threshold value, Cho et al. applied a Markov model to predict next failure time. Above the threshold value, Cho et al. apply a mix of moving average filter and simple linear regression.

Neural networks have showed good performance when applied to time series forecasting in domains other than CBM. Khashei and Bijari (2010) introduced an approach based on using an Auto Regressive Integrated Moving Average (ARIMA) model to extract features from a time series. The features serve as training input to a single hidden layer feedforward network. Khashei and Bijari (2010) test the proposed network on three different time series. Ma et al. (2015) used a long short-term memory (LSTM) neural network to predict traffic speed. The application of time series forecasting methods based on neural networks, combined with huge available amounts of historical data, may lead to better prognostics of industrial machines. Ultimately, better prognostics lead to reduced maintenance costs and increased production availability.

The main goal of this study is to evaluate the performance of a set of methods in the prediction of future values of monitored parameters in industrial machines. The study is an empirical evaluation, done by generating forecasts using a dataset collected from an industrial gas turbine. It evaluates neural network architectures that take as input the entire multivariate time series and output forecasts for all monitored parameters at once. Forecasts for each individual parameter generated by neural networks, ARIMA models, exponential smoothing and naïve models, serve as benchmarks against the multivariate architecture.

The contribution of this paper is in the generation of knowledge directed specifically to the improvement of prognostics, when treated as a time series forecasting problem. To the best of the authors' knowledge, there is a scarce amount of published evaluations of multiple forecasting methods on data from real machines. This knowledge is useful for the understanding of the best forecasting methods available for those who want to estimate remaining useful life of machines.

The structure of the remainder of this study is as follows. Section 2 describes the dataset used in the study and the preprocessing applied to the dataset prior to any model building. Section 2 also describes the methods used to generate forecasts for the time series and the metric for the evaluation of the forecasts. Section 3 presents the results obtained by applying the proposed methods to the dataset. Section 4 provides a conclusion for the study.

THEORETICAL BACKGROUND AND METHODS

Data

This study uses data collected from an oil platform's data historian system. The system stores data from multiple sensors installed through the offshore facility. The focus of the current research is on data collected from the sensors in one of the platform's gas turbines. The turbine operates in power generation role.

The dataset includes values collected from 32 sensors. The dataset contains data from four pressure sensors: P1 is gas fuel pressure, P2 is pressure at the lubrication oil header, P3 is the differential pressure at the inlet air filter and P4 is the pressure at the turbine axial compressor discharge. Two sensors measure rotation: R1 is the Gas Producer (GP) rotor rotation speed and R2 is the Power Turbine (PT) rotor rotation speed. The dataset also contains data from 14 temperature sensors: T1 is temperature at the lubrication oil header, T2 is temperature at the cold junction of the thermocouples installed in the turbine and T3 is the temperature inside the turbine hood. T4 and T8 are oil temperature at inlet of GP and PT rotors bearings respectively, T5, T6 and T9 are the temperatures at the turbine oil sumps, T7 is flow temperature at axial compressor discharge and T10 is the average of the readings of 4 thermocouples installed after the first GP turbine wheel (T11 thru T14). Twelve vibration sensors complete the dataset: V1 thru V10 are radial vibration sensors installed at the turbine's five bearings, V11 is power turbine axial vibration and V12 is auxiliary gearbox casing vibration. Figure 1 shows a schematic of the turbine instrumentation used in this research.

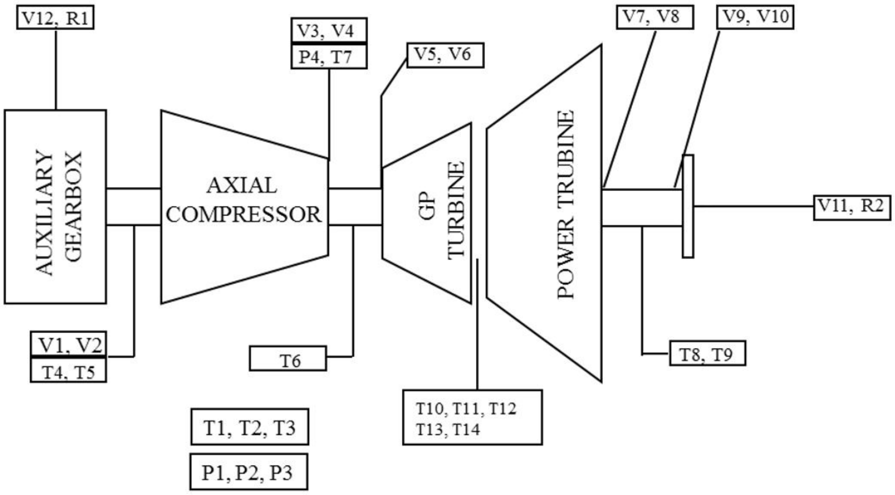

Figure 1 - Position of embedded sensors on the turbine under consideration.
Source: Authors.

Brazilian Journal of Operations & Production Management
Vol. 17, No. 1, e2020607, 2020
DOI: 10.14488/BJOPM.2020.010

The historian system does not log values for each sensor at a fixed sampling rate. Different reasons may trigger a data logging event for each sensor. In order to allow the research to proceed with the use of standard techniques for evenly spaced time series, preprocessing of the dataset aggregates the time series into evenly spaced data. Daily bins divide the dataset. For each time series, the aggregated values equal the mean of the values inside the daily bins. In the event there are no values that fall into a daily bin for one of the series, a linearly interpolated value substitutes the missing value. In order to guarantee secrecy of the real operating parameters, normalization of the dataset using a standard scaler follows. The resulting series have mean zero, standard deviation one and are dimensionless. The resulting dataset contains 1461 daily values for the 32 sensors.

The study splits the dataset in two. The first 90% of the data serves as training data and the last 10% is the test set, used to evaluate the accuracy of the models in proper out of training sample data. The model training phase uses both the original dataset and a second dataset with extreme outliers removed. We define extreme outliers as any value that falls outside of a range defined by a distance of three interquartile ranges from each edge of a boxplot made with all the samples of a time series (Montgomery and Runger, 2010). The use of the dataset without extreme outliers is exclusively for model training. The evaluation of the models trained on the dataset without extreme outliers happens on the regular test set. Figure 2 shows the preprocessed dataset without extreme outliers.

Forecasting Methods

The following subsections describe the forecasting methods applied to the dataset. All methods use the first 90% of data for model training, the definition of model parameters. For all models, the training phase defines the model parameters using one-step ahead forecasts. In the evaluation phase, that uses the remaining 10% of data, there are no changes to the model parameters.

Neural Networks

Neural network is a term that encompass a large class of models and learning methods. Neural networks are nonlinear statistical models that model the outputs as

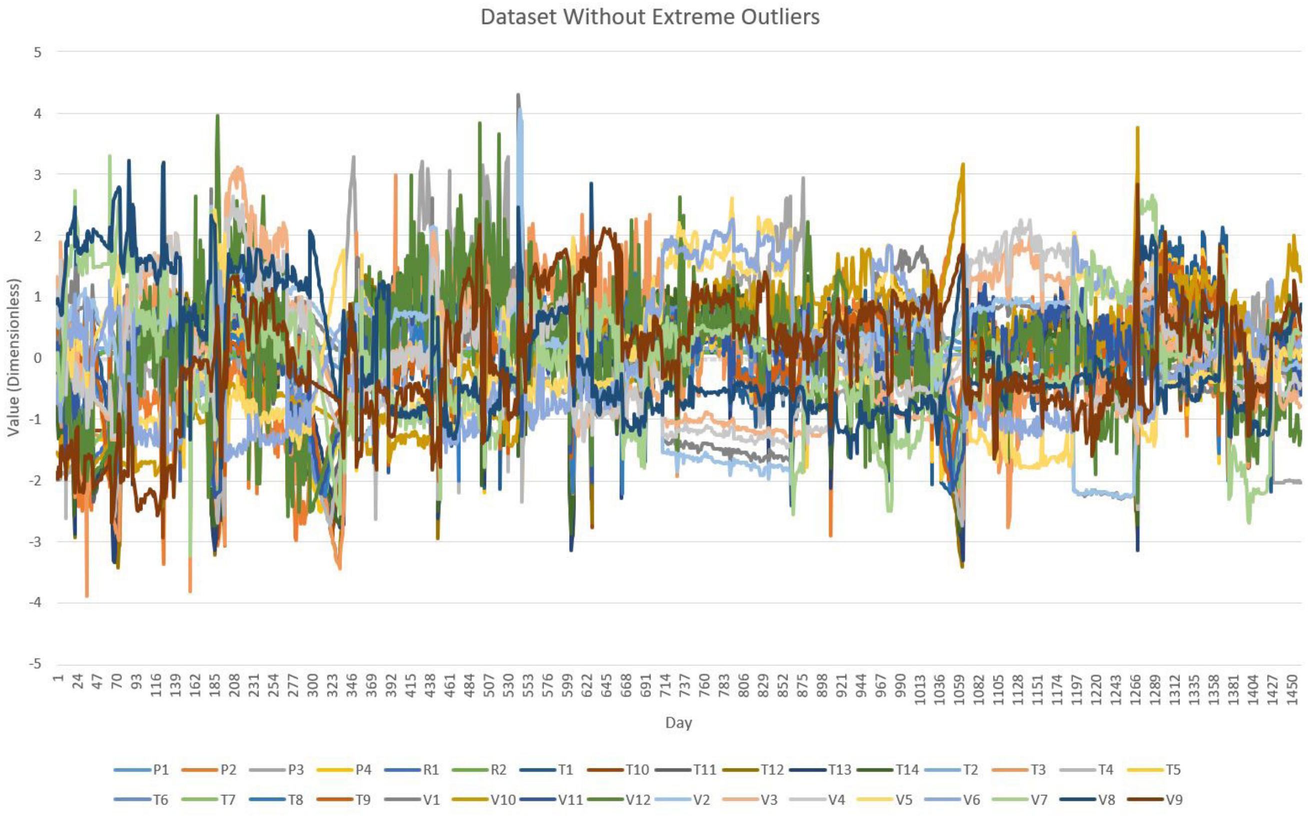

Figure 2 - Dataset without extreme outliers.
Source: Authors.

Brazilian Journal of Operations & Production Management
Vol. 17, No. 1, e2020607, 2020
DOI: 10.14488/BJOPM.2020.010

nonlinear functions of linear combinations of the inputs. One builds a neural network by connecting simple computing cells called neurons or processing units. This study uses neural networks implemented in python using the Keras library (Chollet, 2015).

There are three basic elements to a neuron's model. First, a set of connecting links to other neurons, each characterized by a weight of its own. Second, an adder, often called a propagation function, used to sum all the input signals to the neuron. Third is the activation function. The activation function limits the output of the neuron and is responsible for the nonlinearities in the network. Equation 1 and Equation 2 give the output x_k of a neuron k that uses weighted sum as its adder. Haykin (2009) give further details on the mathematics of neural networks.

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \quad (1)$$

Where:

w_{kj}: weight of the connection between neuron j and neuron k;

x_j: output of neuron j;

b_k: the bias of neuron k.

$$x_k = \varphi(v_k) \quad (2)$$

Where:

v_k: activation potential of neuron k;

φ: the activation function.

One type of neural network used is this study is Feedforward Neural Networks (FNN). There are no loops in an FNN. A layer only uses as input the output from the previous layer. The study tries two approaches to forecasting with FNNs. The first approach is to create a FNN that uses all series as input and outputs the forecasts for the next time step for all of the time series in the dataset at once. We expect that this approach, from now on called multivariate FNN, will be able to capture the interactions between the different time series, resulting in better forecasts. Figure 3 shows an example multivariate FNN. The second approach, called univariate FNN, is to create independent FNNs for each series. The independent FNN only uses as input lagged values from the time series it forecasts. Figure 4 shows an example univariate FNN.

It is necessary to define the architecture of the FNNs before proceeding with the final training. All FNNs used in this study use the same procedure for architecture selection. The procedure starts with a split of the training set: 2/3 for training and 1/3 for validation. Several architectures with one or two hidden layers, different number of units in the hidden layers and different number of lagged values as input are considered. The procedure continues with training of

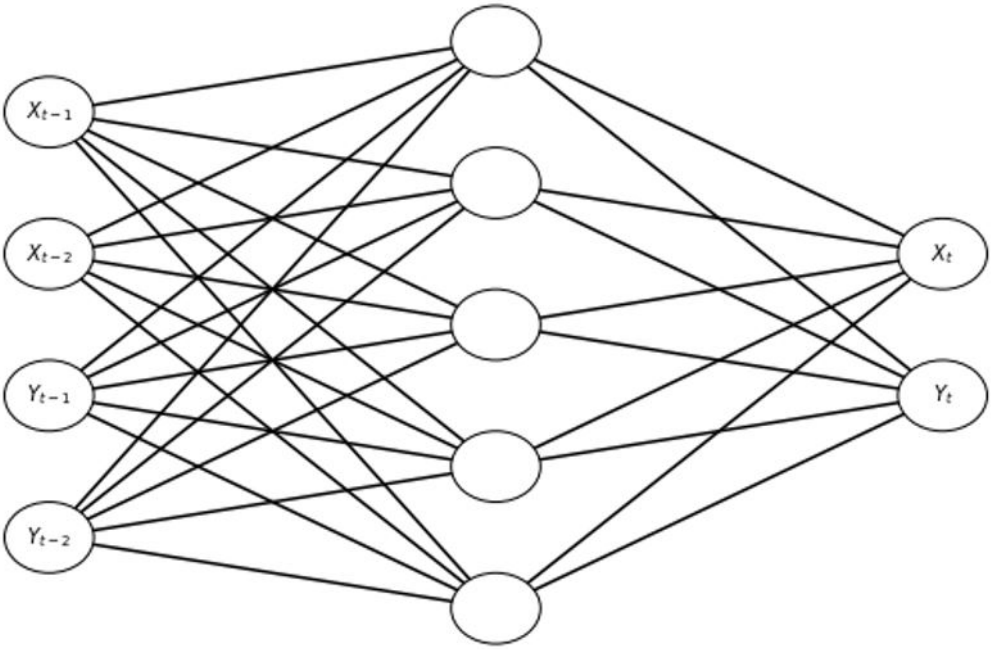

Figure 3 - Multivariate FNN using two lagged values of variables X and Y as input.
Source: Authors.

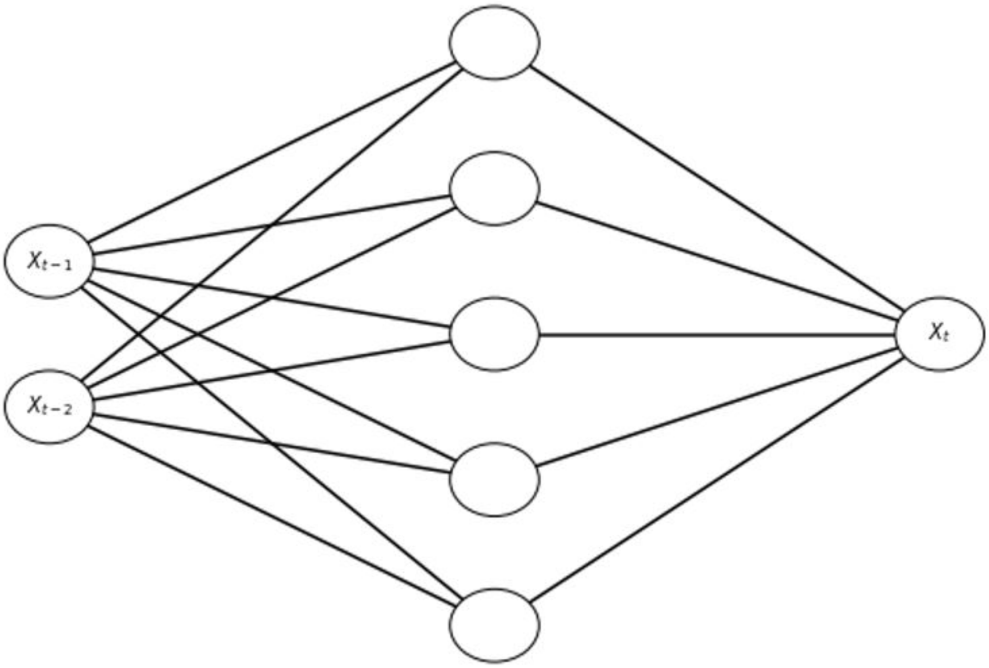

Figure 4 - Univariate FNN using two lagged values of variable X as input.
Source: Authors.

the networks on the reduced training set for a maximum of 1000 epochs, with early stopping if the validation loss (Mean Squared Error (MSE), as in Equation 3) does not improve after 10 consecutive epochs. The final FNNs, trained on the entire training set, use the architectures that presented the smallest average validation loss after 10 training runs. All FNNs use hyperbolic tangent as the activation function of the units in the hidden layer.

$$MSE = \frac{\sum_{t=1}^{k}\left(\hat{y}_t - y_t\right)^2}{k} \tag{3}$$

Where:

$\hat{y}_t$: Forecasted value y at time t;

y_t: Actual value for y at time t;

k: Number of forecasted points.

Long Short Term Memory Units

A Recurrent Neural Network (RNN) is a neural network that allows feedback loops. The state of a RNN, the activations in the hidden units, depends on the past values of the state. The presence of feedback loops makes RNNs specialized in processing sequential data, like time series. RNNs are susceptible to the problem of gradient instability. The longer the network runs, the more unstable are the gradients on inputs further back in time.

Long Short Term Memory (LSTM) units, are a special type of processing unit used to build the hidden layers in a LSTM network. LSTM units address the problem of gradient instability by creating paths through time that have derivatives that will not vanish or explode (Goodfellow et al., 2016). The LSTM units have an adaptive forget gate designed to reset the unit state when its contents are no longer relevant. The forget gate controls the weight of the state self-loop, and in that way, how much of the information in the state is preserved or discarded between time steps. Gers et al. (1999) give further detail on LSTM units.

The study tries two approaches to forecasting with LSTM networks. All LSTM networks tried use only one lagged value as input, since the LSTM units should have the capability to accumulate all relevant past information in their states. The first approach is to create a LSTM that uses all series as input and outputs the forecasts for the next time step for all of the time series in the dataset at once. We expect that this approach, from now on called multivariate LSTM, will be able to capture the interactions between the different time series, resulting in better forecasts. The second approach, called univariate LSTM, is to create independent LSTM networks for each series. The only input of the independent LTSM is the value at $t-1$ of the time series it forecasts.

Brazilian Journal of Operations & Production Management
Vol. 17, No. 1, e2020607, 2020
DOI: 10.14488/BJOPM.2020.010

It is necessary to define the number of LSTM units in the hidden layer before proceeding with the final training. All LSTM networks used in this study use the same procedure to select the size of the hidden layer. The procedure starts with a split of the training set: 2/3 for training and 1/3 for validation. Several architectures with different number of units in the hidden layers are evaluated. The procedure continues with training of the networks on the reduced training set for a maximum of 1000 epochs, with early stopping if the validation loss (MSE) does not improve after 10 consecutive epochs. The final LSTM networks, trained in the entire training set, use the number of hidden units that presented the smallest average validation loss after 10 training runs.

Exponential Smoothing

Exponential smoothing is a forecasting approach that uses all historical values as predictors, giving more weight to more recent values, as in Equation 4 for Simple Exponential Smoothing (SES). The equation shows that the forecast for time $t+1$ is a weighted average between the most recent observation x_t and the forecast for time t. Recursively substituting $\hat{x}_t$ yields Equation 5.

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)(\hat{x}_t) \qquad (4)$$

Where:

$\hat{x}_{t+1}$: Forecasted value for x at time $t+1$;

x_t: Actual value for x at time t;

α: Smoothing parameter.

$$\hat{x}_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + \ldots \qquad (5)$$

As long as $0 < \alpha < 1$, the weight given to each observation decreases exponentially as each observation comes from further in the past, hence the name exponential smoothing. The Holt-Winters procedure, given in the additive form by Equations 6, 7, 8 and 9, generalizes simple exponential smoothing, allowing a trend and a seasonal term. Selecting the best values for the parameters in a Holt-Winters model is a non-linear optimization problem, and the task requires an optimization tool.

$$\hat{x}_{t+h} = l_t + hb_t + s_{t-m+h_m^+} \qquad (6)$$

Where:

b: Trend term;

s: Seasonal term.

$$l_t = \alpha(x_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \qquad (7)$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \qquad (8)$$

$$s_t = \gamma(x_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \qquad (9)$$

Where:

m: Period of seasonality;

α, β, γ: Smoothing Parameters.

This study follows Hyndman et al. (2002) state space approach for ETS (Error, Trend, Seasonality) model selection. An ETS model is a statistical model that underlies an ES method. Each individual time series on the dataset has an ETS model selected for itself. The R function ets() applies Hyndman's approach automatically. It is the computational tool used for ES method selection and parameter estimation.

Auto Regressive Integrated Moving Average

Auto Regressive Integrated Moving Average (ARIMA) models combine Auto Regressive (AR) models, Moving Average (MA) models and differencing. Differencing is a way to make time series stationary by computing the differences between consecutive observations. The addition of differencing allows for non-stationary (on trend) data. An ARIMA (p,d,q) model combines an AR model of order p, a MA model of order q and d order differencing as in Equation 10. The equation uses the backshift operator (B). The backshift operator shifts the data back one period as in equation 11.

$$\left(1 - \varphi_1 B - \ldots - \varphi_p B^p\right)(1-B)^d x_t = C + \left(1 + \theta_1 B + \ldots + \theta_q B^q\right) w_t \qquad (10)$$

Where:

B: Backshift operator;

φ: Auto regressive parameters;

p: Auto regressive order;

θ: Moving average parameters;

q: Moving average order;

d: Differencing order;

w: Error terms;

$$x_t B = x_{t-1} \tag{11}$$

The study uses the R function auto.arima() for model selection and parameter estimation. It conducts a search over possible models and selects the best one based on the smallest Akaike Information Criterion (AIC). Hyndman and Khandakar (2008) give details on the function implementation.

Naïve Forecast

A naïve model is a model that presumes things will remain the same as they have in the past. For time series data, the naïve (no change) model simply forecasts the next observation to be equal as the latest observation. The naïve model serves as a benchmark model for other models. If a model cannot produce better forecasts than a simple alternative like naïve no-change, it is of no use (Armstrong, 2001).

Forecast Evaluation

Forecast accuracy assessment occurs after model training. The metric used for this model evaluation phase is the same used in model training: MSE. The accuracy assessment uses the test set consisting of the last 10% of the full dataset. The model training phase only uses one-step ahead forecast. The model evaluation phase tests model accuracy not only using one-step ahead forecasts, but also using multistep ahead forecasts. Forecasting windows tested are 1, 2, 5, 7, 10 and 14 days ahead. Since the models are configured to produce one step ahead forecasts, a recursive strategy is applied to generate the multistep ahead forecasts. The forecast for time $t+1$ serves as input for the model to forecast the values at time $t+2$. This procedure repeats until the end of the multistep ahead forecast.

Retraining the neural networks after the observation of every new sample in the test set would require significant computational resources. In order to avoid the computational costs, there are no updates to network weights during the model evaluation stage. With the objective of testing the different forecasting methods in the same conditions, there are also no changes to the parameters of the ARIMA and ETS models, even though the computational costs would be significant smaller for these methods.

The calculated MSE are the average for all of the time series. The results do not show what the best method for each univariate time series would be. They show what method would deliver the best results, on average, for a random univariate time series drawn from the dataset, which in turn consists of a diverse collection of time series collected from the same industrial turbine.

RESULTS

The model that uses a one hidden layer feedforward neural network taking as input all of the time series and trying to predict the next values for all series requires selection of hyperparameters before training. The hyperparameters selection phase uses a split of the training set: 2/3 for training and 1/3 for validation. Table 1 summarizes all the hyperparameters considered. Table 2 shows the average validation loss for each of the architectures after 10 training runs. Based on these results, the final model uses 180 units in the hidden layer and a single time lag for the inputs.

The model with two hidden layers that predicts all values simultaneously uses the same 2/3 for training and 1/3 for validation split for selection of hyperparameters, as in the one hidden layer model. This step considered the same possible time lags and units in the hidden layers considered for the one hidden layer model. Table 3 shows the average

Table 1 - Considered architectures for the one hidden layer feedforward neural network.

Time Lags	1, 2, 3, 4, 5
Hidden Layer Units	1, 2, 5, 10, 20, 33, 50, 66, 100, 133, 150, 180, 200

Source: Authors.

Table 2 - Average validation loss for the considered one hidden layer FNN architectures.

Time Lags	**Hidden Layer Units**										
	1	**2**	**5**	**10**	**20**	**50**	**100**	**133**	**160**	**180**	**200**
1	0.76393	0.62184	0.50953	0.43723	0.32314	0.25762	0.23207	0.23139	0.22623	0.21628	0.22946
2	0.75941	0.62024	0.53573	0.45008	0.38593	0.28650	0.26639	0.26157	0.26087	0.26132	0.25844
3	0.76960	0.64555	0.55353	0.47113	0.38620	0.32633	0.29821	0.29330	0.29665	0.29493	0.28769
4	0.79241	0.69564	0.56170	0.49333	0.39378	0.33650	0.32520	0.32919	0.32796	0.30325	0.32996
5	0.81911	0.67711	0.56336	0.48584	0.40795	0.33522	0.33197	0.34921	0.33123	0.32149	0.33314

Source: Authors.

Brazilian Journal of Operations & Production Management
Vol. 17, No. 1, e2020607, 2020
DOI: 10.14488/BJOPM.2020.010

Table 3 - Average validation loss for the top 10 architectures of two hidden layers FNN.

Time Steps	Layer-1 Units	Layer-2 Units	Validation Loss
1	160	180	0.22180
1	200	200	0.22607
1	160	200	0.23345
1	180	100	0.23773
1	180	160	0.23824
1	200	180	0.24917
1	200	160	0.25318
2	100	180	0.25329
2	133	180	0.25388
1	133	133	0.25541

Source: Authors.

Table 4 - Average validation loss for the considered LSTM architectures.

Hidden Layer Units	Validation Loss
1	0.97163
2	0.71791
5	0.64105
10	0.56996
20	0.53536
50	0.48890
100	0.56135
133	0.63046
160	0.69596
180	0.66378
200	0.70038

Source: Authors.

validation loss for the 10 best architectures after 10 training runs. Based in these results, the final model uses 160 units in the first hidden layer, 180 units in the second hidden layer and a single time lag for the inputs.

The multivariate LSTM model considered uses only one hidden layer. The input vector uses only one lagged value from all the series since the states of the LSTM units should be able to save any relevant information on values of the time series further back in time. The only hyperparameter selected for the network is the number of units in the hidden layer. Table 4 shows the average validation loss for each of the architectures after 10 training runs. Based on these results, the final model uses 50 units in the hidden layer.

The prediction methods that use independent networks for each time series use a hyperparameter selection procedure similar to the one described for the architectures that predict all of the time series simultaneously. For the individual time series, we do not consider two hidden layers FNN architectures. Table 5 shows the architectures considered for each independent FNN. The independent LSTM networks tested use one time lag as input and the same number of units in the hidden layer as the considered FNNs. Table 6 shows the selected FNN architectures and Table 7 shows the selected LSTM architectures for each univariate time series, based on the average validation loss after 10 training runs.

Table 5 - Architectures considered for each independent neural network.

Time Lags	1, 2, 3, 4, 5
Hidden Layer Units	1, 2, 3, 4, 5, 8, 10, 15, 20

Source: Authors.

Table 8 and Table 9 summarize the results obtained by applying all the proposed forecasting methods to the test set. Table 8 shows the MSE for the scenario with models trained on the normal training set (with extreme outliers). Table 9 shows the MSE for the scenario with models trained on the training set without extreme outliers. The comparison between the two tables show that in general (26 out of 36 cases) removing the extreme outliers improves the forecast. However, all instances where the removal of the extreme outliers worsened the forecast happened on the methods of better performance (univariate LSTM, ARIMA and ES).

The forecasting method that showed the worst performance was the multivariate FNN. To the best of the authors' efforts, it was not possible to design and train a multivariate FNN capable of matching or surpassing the performance of the other methods, included the use of univariate FNNs. The multivariate FNNs performs worse than a simple naïve method in all forecast windows when trained on the normal training set. When trained on the training set without extreme outliers, the two hidden layers FNN is capable of beating the naïve model in three of the six forecast windows tested. The univariate FNN beats the naïve forecast in forecast windows equal or bigger than two.

Brazilian Journal of Operations & Production Management
Vol. 17, No. 1, e2020607, 2020
DOI: 10.14488/BJOPM.2020.010

Table 6 - Selected FNN architecture for each univariate time series.

	P1	P2	P3	P4	R1	R2	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9	V1	V10	V11	V12	V2	V3	V4	V5	V6	V7	V8	V9
Hidden Layer Units	3	8	10	10	1	3	5	8	8	15	5	8	15	8	15	20	10	4	8	10	15	20	15	5	20	15	20	15	15	15	15	5
Time Lags	1	4	1	1	5	3	3	2	2	5	2	2	1	1	3	3	2	1	2	3	2	2	3	3	2	1	2	3	1	3	1	2

Source: Authors.

Table 7 - Selected LSTM architecture for each univariate time series.

	P1	P2	P3	P4	R1	R2	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9	V1	V10	V11	V12	V2	V3	V4	V5	V6	V7	V8	V9
Hidden Layer Units	20	5	3	8	20	20	8	8	5	4	8	10	20	4	20	15	20	4	15	15	15	20	20	1	2	1	1	1	1	2	5	1

Source: Authors.

Table 8 - MSE for each of the forecasting methods using the normal training set.

Days in Forecast Window	Normal Dataset Test Loss - Mean Squared Error							
	FNN1 Mult	FNN2 Mult	LSTM Mult	FNN Uni	LSTM Uni	ARIMA	ES	Naive
1	0.31949	0.31099	0.34494	0.20698	0.20318	0.16538	0.18055	0.17894
2	0.40082	0.39742	0.39136	0.25457	0.24811	0.22355	0.25041	0.25484
5	0.63193	0.67098	0.52512	0.35353	0.34054	0.33863	0.40570	0.43541
7	0.81206	0.86482	0.60214	0.39451	0.37599	0.37346	0.45626	0.49433
10	1.24430	1.16320	0.72377	0.45896	0.42069	0.40702	0.50454	0.54906
14	2.10739	1.53733	0.91939	0.54004	0.47653	0.43645	0.54369	0.59119

Source: Authors.

Table 9 - MSE for each of the forecasting methods using the training set without extreme outliers.

Days in Forecast Window	Dataset Without Extreme Values Test Loss - Mean Squared Error							
	FNN1 Mult	FNN2 Mult	LSTM Mult	FNN Uni	LSTM Uni	ARIMA	ES	Naive
1	0.27152	0.25390	0.34341	0.20294	0.20947	0.17585	0.18200	0.17894
2	0.32619	0.30878	0.39133	0.25100	0.25354	0.22904	0.25318	0.25484
5	0.44644	0.42141	0.49046	0.35396	0.34265	0.33247	0.41182	0.43541
7	0.50626	0.47053	0.54292	0.40036	0.37726	0.36285	0.46296	0.49433
10	0.59500	0.54547	0.63003	0.46940	0.41971	0.39086	0.51011	0.54906
14	0.68721	0.64024	0.72755	0.54970	0.46524	0.41600	0.54840	0.59119

Source: Authors.

The use of LSTM units generally improves the performance of the neural networks. Similar to the FNN case, using univariate LSTM networks yields better results than using a multivariate LSTM. In this study, attempting to capture the interactions between the different time series did not produce better forecasts. For forecast windows bigger than one, the univariate LSTM is capable of delivering better results than the no-change naïve method.

Exponential smoothing applied to each individual time series showed the third best performance on the study. ES only produces worse forecasts than the naïve method in the one-step ahead forecast scenario, but loses to univariate LSTM networks in forecasting windows bigger than 2. The winning method in the evaluation is the use of ARIMA models, selected for each individual time series, trained on the original training set (with outliers), using Hyndman's approach for model order selection. The use of ARIMA models consistently beats the naïve forecasts.

The results found by the study show, based on the dataset explored, that should one select a single method, between the ones tested in this study, to produce forecasts on a set of time series drawn from industrial machinery sensors, the default forecasting method should be the use of independent ARIMA models selected for each univariate time series.

Not only ARIMA models produced the best forecasts in this empirical evaluation, but also the computational resources required to select the model parameters are much smaller.

CONCLUSION

This work empirically evaluated the forecasting performance of a set of different forecasting methods in a dataset of industrial sensors time series. The dataset comes from an oil platforms' data historian system. The results show that using ARIMA models to forecast these time series is the best default methodology to apply, and is the only methodology that consistently beats a simple naïve no-change model.

This study presents limitations. There was a focus on evaluating neural networks. This limited the time available to evaluate other forecasting methods developed by the computer science (e.g., Support Vector Regression) and statistics communities. Moreover, there is no guarantee that it would not be possible to find neural networks capable of yielding better forecasts than the ones achieved by the best performing methods in this research. Other limitation is that while the dataset consisted of 32 time series, these were drawn from only 4 types of machine sensors.

Future studies should focus on improving the variety of the time series in the dataset, assessing a greater variety of forecasting methods and finding better performance of models based on neural networks.

REFERENCES

Alarcón, F., Perez, D. and Boza, A. (2016), "Using the internet of things in a production planning context", *Brazilian Journal of Operations & Production Management*, Vol. 13, No. 1, pp. 72-6. https://doi.org/https://doi.org/10.14488/BJOPM.2016.v13.n1.a8

Armstrong, J.S. (2001), *Principles of Forecasting: A Handbook for Researchers and Practitioners*, 1st ed., Springer Science & Business Media, Norwell, MA.

Cho, S., Shin, J.-H., Jun, H.-B. et al. (2016), "A study on estimating the next failure time of compressor equipment in an offshore plant", *Mathematical Problems in Engineering*, Vol. 2016, available at: https://www.hindawi.com/journals/mpe/2016/8705796/abs/ (accessed 2 October 2018).

Chollet, F. (2015), "Keras", available at: https://keras.io (accessed 2 October 2018).

Crone, S.F., Hibon, M. and Nikolopoulos, K. (2011), "Advances in forecasting with neural networks: empirical evidence from the nn3 competition on time series prediction", *International Journal of Forecasting*, Vol. 27, No. 3, pp. 635-60.

Datong, L., Yu, P. and Xiyuan, P. (2011), "Online adaptive status prediction strategy for data-driven fault prognostics of complex systems", in *Prognostics and System Health Management Conference*, Shenzhen, China, pp. 1-6.

Dragomir, O.E., Gouriveau, R., Dragomir, F. et al.(2009), "Review of prognostic problem in condition-based maintenance", in *2009 European Control Conference (ECC)*, IEEE, Budapeste, Hungary, pp. 1587-1592.

Gers, F.A., Schmidhuber, J. and Cummins, F. (1999), *Learning to Forget: Continual Prediction with LSTM*, IDSIA, Switzerland.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), "Deep Learning: An MIT Press book", available at: http://www.deeplearningbook.org (accessed 2 October 2018).

Haykin, S. (2009), *Neural Networks and Learning Machines*, 3rd ed., Pearson, Upper Saddle River, NJ.

Heng, A., Tan, A.C., Mathew, J. et al. (2009), "Intelligent condition-based prediction of machinery reliability", *Mechanical Systems and Signal Processing*, Vol. 23, No. 5, pp. 1600-14.

Hyndman, R.J. and Khandakar, Y. (2008), "Automatic time series forecasting: The forecast package for R", *Journal of Statistical Software*, Vol. 26, No. 3, pp. 1-22.

Hyndman, R.J., Koehler, A.B., Snyder, R.D. et al. (2002), "A state space framework for automatic forecasting using exponential smoothing methods", *International Journal of Forecasting*, Vol. 18, No. 3, pp. 439-54.

Khashei, M. and Bijari, M. (2010), "An artificial neural network (p, d, q) model for timeseries forecasting", *Expert Systems with Applications*, Vol. 37, No. 1, pp. 479-89.

Lee, J., Wu, F., Zhao, W. et al. (2014), "Prognostics and health management design for rotary machinery systems -- reviews, methodology and applications", *Mechanical Systems and Signal Processing*, Vol. 42, No. 1, pp. 314-34.

Lopes Miranda Junior, H., Albuquerque Bezerra, N., Soares Bezerra, M. et al. (2017), "The internet of things sensors technologies and their applications for complex engineering projects: a digital construction site framework", *Brazilian Journal of Operations & Production Management*, Vol. 14, No. 4, pp. 567-76. available at: https://doi.org/https://doi.org/10.14488/BJOPM.2017.v14.n4.a12 (accessed 12 January 2019).

Ma, X., Tao, Z., Wang, Y. et al. (2015), "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data", *Transportation Research Part C, Emerging Technologies*, Vol. 54, pp. 187-97.

Montgomery, D.C. and Runger, G.C. (2010), *Applied Statistics and Probability for Engineers*, 5th ed., John Wiley & Sons, Hoboken, NJ.

Niu, G. and Yang, B.-S. (2010), "Intelligent condition monitoring and prognostics system based on data-fusion strategy", *Expert Systems with Applications*, Vol. 37, No. 12, pp. 8831-40.

Pham, H.T., Yang, B.-S., Nguyen, T.T. et al.(2012), "Machine performance degradation assessment and remaining useful life prediction using proportional hazard model and support vector machine", *Mechanical Systems and Signal Processing*, Vol. 32, pp. 320-30.

Qin, S.J. (2014), "Process data analytics in the era of big data", *AIChE Journal. American Institute of Chemical Engineers*, Vol. 60, No. 9, pp. 3092-100.

Shin, J.-H. and Jun, H.-B. (2015), "On condition based maintenance policy", *Journal of Computational Design and Engineering*, Vol. 2, No. 2, pp. 119-27.

Received: 30 Oct 2018
Approved: 19 Jul 2019

How to cite: Santos, H.F.R., Silva, L.W.C. and Sobral, A.P.B. (2020), "Forecast of multivariate time series sampled from industrial machinery sensors", *Brazilian Journal of Operations & Production Management*, Vol. 17, No. 1, e2020607. https://doi.org/10.14488/BJOPM.2020.010