Sensor Data Preprocessing, Feature Engineering and Equipment Remaining Lifetime Forecasting for Predictive Maintenance

© Evgeniy Latyshev
Lomonosov Moscow State University,
Moscow, Russia
e.latishev@gmail.com

Abstract. Analytics based on sensor data is gradually becoming an industry standard in equipment maintenance. However, it involves several challenges, such as sensor data preprocessing, feature engineering and forecasting model development. Due to work in progress, this paper is mainly focused on sensor data preprocessing, which plays a crucial role in predictive maintenance due to the fact, that real-world sensing equipment usually provides data with missing values and a considerable amount of noise. Obviously, poor data quality can render practically useless all the following steps of data analysis. Thus, many missing data imputation, outlier filtering, and noise reduction algorithms were introduced in the literature. Streaming sensor data can be represented in a form of univariate time series. This paper provides an overview of common univariate time series preprocessing steps and the most appropriate methods, with consideration of the field of application. Sensor data from different sources comes in different scales and should be normalized. Thus, the comparison of univariate time series normalization techniques is given. Conventional algorithm quality metrics for each of the preprocessing steps are described. Basic sensor data quality assessment approach is suggested. Moreover, the architecture of a sensor data preprocessing module is proposed. The overview of time series-specific feature engineering techniques is given. The brief enumeration of considered forecasting approaches is provided.

Keywords: predictive maintenance, preprocessing, univariate time series, data cleaning, missing data imputation, noise reduction, outlier filtering, data quality assessment, feature engineering, time series forecasting

1 Introduction

Maintenance costs are a major part of the total operating costs of any business involving complex equipment. Conducted surveys of maintenance management effectiveness indicate that one-third of all maintenance costs is wasted as the result of unnecessary or improperly carried out maintenance [16]. With the spread of Internet of Things concept, sensor data can be collected from a huge amount of devices and equipment. This data can be used for real-time health monitoring and effective maintenance. However, this approach to maintenance, also known as predictive maintenance, involves several challenges.

First of all, often collected data is of poor quality, which can lead to unreliable analysis and ineffective maintenance. Consequently, data from sensing equipment needs to be preprocessed before it can be used for any analysis. Poor data quality means non-compliance with requirements on at least one of the data quality assessment metrics. The root of problems can vary: connection issues, sensor malfunction, transmitting hardware failure, data processing server downtime, software crash, measuring equipment inaccuracy and many more. Common cases of poor data quality involve unacceptable amount of missing values, outliers, sudden spikes etc. Simply ignoring these issues can be critical due to several reasons. For example, some analysis tools, including popular machine learning algorithms, can't handle missing values. The absence of outlier filtration can dramatically skew the results. Measuring equipment standard error can be mistaken for an actual pattern in data. As a result, time series preprocessing involves several independent steps: missing data imputation, noise reduction, and data normalization. After these steps, data can be evaluated in quality and passed further for analysis. It is clear, that preprocessing should be done in near real-time to minimize the delay between data measurement and decision making. Thus, there is a need for a fast and scalable independent module, that can preprocess constantly incoming sensor data. This paper proposes the design of such a module, keeping in mind the following integration with the existing architecture of a predictive maintenance system, introduced in [14].

Secondly, it can be difficult to distinguish the patterns and relationships in the initial data. The process of extracting and generating new characteristics and features out of the available data, commonly referred to as feature engineering, has two main objectives. The first one is to represent the data in such a form, that will make it easier to establish simple yet strong connections between the input and the output variables for the forecasting model, increasing the quality of the forecasts.

Proceedings of the XX International Conference "Data Analytics and Management in Data Intensive Domains" (DAMDID/RCDL'2018), Moscow, Russia, October 9-12, 2018

The second objective is to pick the most useful features out of all the available ones, reducing the amount of computations of the forecasting model.

Finally, a proper forecasting model is to be chosen and implemented. There are various approaches to time series forecasting, from straightforward methods like naive method to way more sophisticated ones like long-short term recurrent neural network. The main complication here is the trade-off between the forecast quality and the ease of model implementation and deployment.

The remaining part of the paper is organized as follows. The preprocessing module architecture is described in Section 2. Section 3 reviews missing data imputation methods. Section 4 is devoted to time series noise reduction. Section 5 briefly overviews data normalization techniques. In section 6, some thoughts on data quality assessment are combined. Section 7 is devoted to time series feature engineering. The brief overview of time series forecasting approaches is given in section 8. Finally, the future directions of presented work are given in section 9.

2 Preprocessing Module Architecture

The preprocessing module is a part of the system for predictive maintenance, deployed to a Hadoop [29] cluster in a cloud manner. The module is wrapped in Docker [17] container and runs on a standalone node of the cluster. One of the key requirements for the module is the seamless integration into the architecture. The data is retrieved from Apache Kafka message queue [2], transformed by the preprocessing module and passed in parallel to OpenTSDB [25] and Apache Hive [11] for storage. To satisfy the requirements onto speed and scalability the transformations are conducted onto Apache Spark Streaming engine [27].
There are many stream data processing frameworks including but not limited to Apache Storm [5], Apache Flink [1], Apache Samza [4] and Kafka Streams [3]. Although Spark Streaming has latency issues and sliding window processing may be tricky due to Spark inherent batch-based streaming model, is has several advantages which make Spark Streaming a safer choice.
First of all, Spark Streaming is a mature framework with thorough documentation and huge community. As a result of long-term popularity, there are plenty of open-source tools for Spark Streaming, including solutions for relatively painless integration with Kafka and mentioned earlier database management systems [28, 11, 21].
Another advantage is the existence of pySpark [23] – an API for Python, one of the biggest programming languages at this moment. All other enumerated frameworks require Scala, Clojure or Java knowledge, which makes them less accessible.
One of the biggest downsides of Spark Streaming is performance degradation on sudden bursts of input data. However, in case of sensor data processing the input data flow intensity remains nearly the same at all time intervals, which mitigates the downside.

The data flow and module components are introduced below in figure 1. The whole module consists of 4 transformation steps and the data quality assessment step.

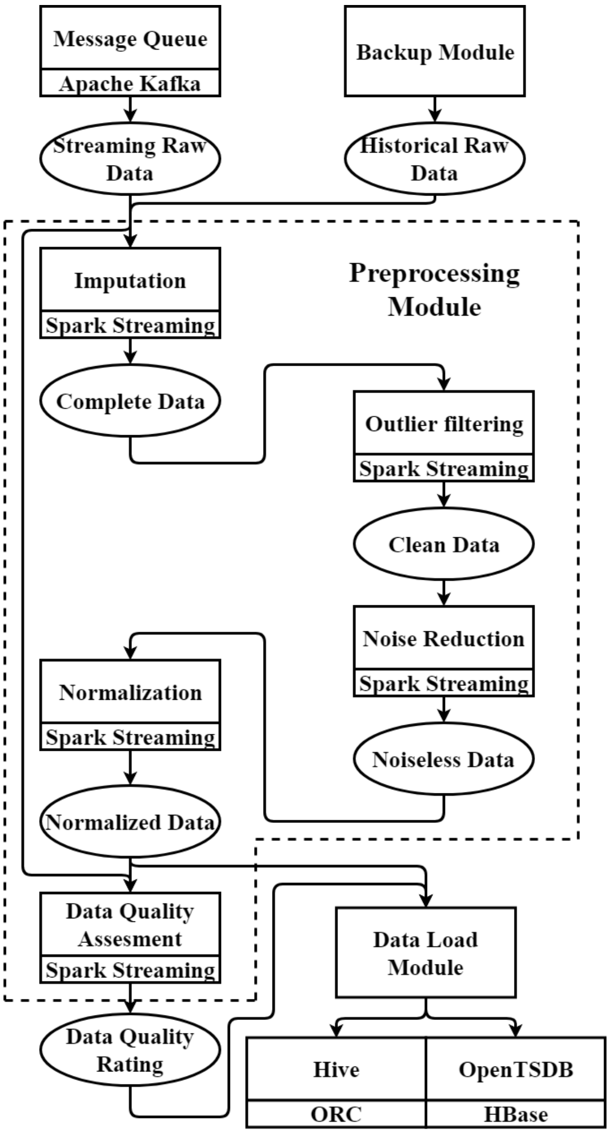

Figure 1 Components and data flow within the preprocessing module

3 Missing Data Imputation

Sometimes due to a sensor malfunction, unstable internet connection or other technical difficulties the data for some points in time is missing. Simply ignoring those gaps may be not the best strategy, because it can lead to a loss of efficiency and unreliable results of the analysis. Another approach is to try to impute the missing values based on the available information.

3.1 Methods

The detailed overview of basic imputation methods and their implementations can be found in imputeTS R package documentation [19].

Some simple methods that are applicable not only to time series: median imputation, mode imputation, mean imputation, random imputation. These methods are fast and very straight-forward, but lack accuracy.

Simple time series specific methods include LOCF (last observation carried forward), NOCB (next observation carried backward), interpolation (linear, polynomial, Stineman) and moving average (simple, weighted, exponential). All of them are rather fast and can work in specific cases, but fall off when there is seasonality in the data or large missing sub-sequences.

More sophisticated approaches like Structural Model & Kalman Smoothing, ARIMA State Space Representation & Kalman Smoothing [10] can be used for seasonal data with complex patterns.

However, sensor data has one unfortunate characteristic – the gaps of missing data can be too long for conventional methods to work properly. In this case, the method proposed in [22] can be the appropriate choice. The idea of the Dynamic Time Warping Based Imputation is to find the most similar sub-sequence to the sub-sequence before the missing values, then complete the gap by the next sub-sequence of the most similar one. The result is a very plausible gap imputation with a drawback of a huge computational cost.

3.2 Metrics

Missing data imputation involves 2 types of quality metrics based on the pattern of imputation.

For single value imputations, the metrics coincide with the ones commonly used in time series forecasting – RMSE (Root Mean Square Error)and MAPE (Mean Absolute Percentage Error).

$$RMSE = \sqrt{\frac{\sum_i (\hat{y}_i - y_i)^2}{n}},$$

$$MAPE = \frac{100\%}{n} \times \sum_i \left| \frac{y_i - \hat{y}_i}{y_i} \right|,$$

where y_i is real value, $\hat{y}_i$ is the forecasted value and n is the number of forecasts.

However, different metrics are used for long gap imputation. The most popular of them are similarity and Dynamic Time Warping distance.

$$Similarity = \frac{1}{n} \times \left(\sum_i \frac{1}{\left(1 + \frac{|y_i - \hat{y}_i|}{max(\hat{y}_i) - min(\hat{y}_i)}\right)} \right)$$

DTW calculation algorithm can be found at [19]. It is worth mentioning, that modern implementations often have adjustments to speed up the calculations (for example, DDTW [13]).

4 Noise Reduction

Similar to missing data points, sensor data is usually contaminated with noise, which can be mistaken for actual data pattern, which yet again leads to a loss of efficiency and unreliable results of the analysis. The task of noise reduction is to subtract the maximum amount of noise from the initial data, leaving the maximum amount of useful signal.

4.1 Methods

According to Chen et al. [6] noise reduction methods can be divided into 2 categories: frequency domain approaches and time domain approaches.

Frequency domain approaches are based on signal decomposition into frequency components. The most common approaches involve discrete/fast/short-time Fourier transform either wavelet transform.

Most of the time domain approaches are based on smoothing the signal of each given data point based on the values of its neighbors.

The comparison of the basic noise reduction methods can be found in the work of Köhler et al. [15]. The conducted experiment involves the comparison of moving average filter, exponential smoothing filter, linear Fourier smoothing, nonlinear wavelet shrinkage and simple nonlinear noise reduction in different conditions.

The downside of the approaches listed above is that they modify almost all the data values, most of which are initially correct. Song et al. [26] proposed the first constraint-based approach for cleaning stream data. The idea is to sanity check the changes of values in time based on subject area constraints. This method allows to detect and repair large spike errors in data. The biggest advantage of this method is the support of online cleaning over streaming data.

However, this method can be used only for large outlier detection. In some cases, even small errors can be important and repairing only spike errors is insufficient. Zhang et al. [30] proposed a novel statistical-based cleaning by introducing the repairment likelihoods with respect to speed changes. Several effective and computationally efficient heuristics are also introduced in this work.

4.2 Metrics

Most of the papers involve RMSE, defined earlier, as a denoising quality metric. However, there are several less popular ones, including the Symmetrical Visual Error Measure, proposed in [18].

5 Data Normalization

Making sure that your data is of uniform scale is key for many methods, including k-NN, linear models, artificial neural networks and many more. Even univariate time series data should be normalized because it might be further used in combination with data of different scale from other sources.

The most well-known and widely used are min-max normalization and z-score normalization. Min-max implies that you know the minimum and the maximum values in your dataset beforehand, which is often not the case. Z-score is more robust but performs poorly om non-stationary time series.

$$\hat{y}_{minmax} = \frac{y - min(Y)}{max(Y) - min(Y)},$$

$$\hat{y}_{z-score} = \frac{y - mean(Y)}{std(Y)},$$

where ŷ is a value after normalization, y is a value prior normalization and Y is the set of values being normalized.

Some less popular methods are decimal scaling normalization, which holds all the drawbacks of min-max normalization, sigmoid normalization, which is actively used in neural networks and tanh estimators, which roughly can be described as a hyperbolic tangent of the z-score normalization.

$$\hat{y}_{decimal} = \frac{y}{10^{d}},$$

where d is the order of values in the set.

$$\hat{y}_{sigmoid} = \frac{1}{1+e^{y}},$$

$$\hat{y}_{tanh} = 0.5 * \left(tanh\left(\frac{0.01*(y-mean(Y))}{std(Y)}\right) + 1\right).$$

According to the experiment, conducted in [20], there is no optimal time series normalization method and one should choose the appropriate method based on the data patterns. Regarding sensor data, the mean and standard deviation remain approximately the same throughout time, which makes z-score normalization a reasonable choice.

6 Data Quality Assessment

Data Quality Assessment (DQA) is the scientific and statistical evaluation of data to determine if data obtained from environmental data operations are of the right type, quality, and quantity to support their intended use [8].

There is a comprehensive work on time series data quality assessment done in [9], which shows that there are dozens of different metrics that can be used to measure the quality of data. Obviously, using all of them is excessive and computationally inefficient, so only a few are to be chosen. However, there is no common view on which metrics are better. The simple yet effective strategy might be to look onto the most popular ones:

- event data loss (gaps in the data);
- values out of range (values out of sane interval for the domain);
- value spikes (improbable sudden changes);
- wrong timestamps;
- rounded measurement value (not desirable level of detail);
- signal noise (slightly inaccurate measurements).

The assessment is to be done for both data prior and after preprocessing to acquire an evaluation of preprocessing module effectiveness. It is also worth keeping in mind, that initially clean data is different to the data, that was made "clean" during preprocessing due to approximations and inevitable errors of the methods involved on each step.

7 Feature Engineering

Feature engineering is, probably, the most peculiar step of data processing, as it depends on the initial data type, its origin, quantity, quality, the desired output of the forecasting model and even the nature of the model itself. As it was already mentioned, sensor data can be represented in a form of univariate time series. The conventional approaches to time series feature engineering can be divided into 3 categories: timestamp features, statistical features, and spectral features. The feature extraction step is usually followed by a dimensionality reduction step.

It is worth mentioning, that there are automatic time series feature engineering tools such as tsfresh [7], which achieve decent results with almost no effort required.

7.1 Timestamp Features

The idea of this approach is to extract the features from the timestamp of each observation. The most commonly used features are:

- minutes elapsed for the day;
- hour of the day;
- day of the month;
- weekend or not;
- season of the year;
- public holiday or not.

Talking about sensor data, some examples of useful timestamp features are:

- time since the last maintenance;
- age of equipment;
- time since the last failure;
- operating time of equipment.

Using just these features alone for predictions will likely result in a poor model. However, in combination with other features, they can boost the quality of forecasts.

7.2 Statistical Features

This approach involves sliding through a time series with the window of a given width and calculating statistics for each iteration. The most common statistical features are the mean of the previous few values, the median, the mode, the minimal value, the maximum value, the standard deviation and many more. In addition to calculated statistics, we can also use the lagged values of a time series as features.

The biggest challenge of this approach is that the window can be of any width and there is no general algorithm to choose it. Usually, the researchers just try out several values of the width and choose the one that performs best. However, if there is a seasonal pattern in data, it is worth making the width of the window not less than the period of the seasons.

7.3 Spectral Features

Different variations of Fourier transform and wavelet transform are used to extract spectral features from a nonstationary signal. The basic idea behind those methods is to decompose a given time series into a sum of several basic functions, providing a different representation of the initial signal. The biggest drawback of those methods is that they are relatively computationally expensive.

7.4 Dimensionality Reduction

Feature extraction provides many features, some of which can be useless or strongly correlated with each other. Excessive features not only add unnecessary computations but also can decrease the quality of the model. Thus, several dimensionality reduction methods were introduced to minimize the number of features, at the same time keeping the maximum amount of information. The most common ones are principal component analysis, independent component analysis and partial least squares regression.

8 Remaining Lifetime Forecasting

There is a variety of methods, that can be used for time series forecasting. Each of them has advantages and drawbacks and can be viable in certain circumstances.

First of all, there are some basic methods such as average method, naive method, seasonal naive method, and drift method, which are very simple yet can be effective when the data pattern is easy.

Secondly, linear regression models can be used for forecasting. In the simplest case, the regression model allows for a linear relationship between the forecast variable and some predictor variables. The biggest downside is inherent linearity, while real-world data is mostly non-linear.

The most common approach is to use stochastic models – ARMA, ARIMA, SARIMAX, etc. One of the biggest drawbacks of those models is that they require fine-tuning of several hyperparameters, which is computationally expensive and not intuitive.

One of the recently popular approaches is to use decision trees. Random forests and gradient boosting methods, which are so widely used in machine learning competitions, can also be used for time series forecasting.

Artificial neural networks approach for time series forecasting gained immense popularity in last few years. Their modifications – recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) are especially effective for this task due to their "memory" component. Although neural networks tend to be the most accurate forecasting method when tuned properly and given enough data, they might be computationally too costly for the considered conditions.

9 Conclusion

In this study, the overview of sensor data preprocessing steps, methods, and common metrics is held. Some thoughts on sensor data quality assessment are given. The architecture of a fast, scalable preprocessing module is proposed. A brief overview of time series feature engineering techniques and forecasting methods is given. The future goals of the ongoing work are to implement the designed preprocessing module on Spark Streaming engine, integrate it into the existing predictive maintenance pipeline, implement the feature engineering step and to develop a remaining lifetime forecasting model.

Acknowledgments. This work is supervised by Dmitriy Kovalev, Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences.
The research is financially supported by Ministry of Education and Science of the Russian Federation (project's unique identifier RFMEFI60717X0176).

References

[1] Apache Flink. https://flink.apache.org/

[2] Apache Kafka. https://kafka.apache.org/

[3] Apache Kafka Streams Documentation. https://kafka.apache.org/documentation/streams/

[4] Apache Samza. http://samza.apache.org/

[5] Apache Storm. https://storm.apache.org/

[6] Chen, Mithal, Vangala, Brugere, Boriah, Kumar: A study of time series noise reduction techniques in the context of land cover change detection. NASA Conference on Intelligent Data Understanding (2011)

[7] Christ M., Kempa-Liehrb A., Feindt M.: Distributed and Parallel Time Series Feature Extraction for Industrial Big Data Applications. ACML Workshop on Learning on Big Data (2016)

[8] Gitzel R.: Data Quality in Time Series Data An Experience Report. CBI Industrial Track (2016)

[9] Guidance for Data Quality Assessment: Practical Methods for Data Analysis. EPA (2000)

[10]Harvey A.: Forecasting, structural time series models and the Kalman filter. Cambridge university press (1990)

[11]Hive on Spark: Getting Started. https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark%3A+Getting+Started

[12]Huai Y., Chauhan A., Gates A., Hagleitner G., Hanson E.N., O'Malley O., Pandey J., Yuan Y., Lee R., Zhang X.: Major technical advancements in Apache Hive. ACM SIGMOD international conference on management of data (2014)

[13]Keogh, E., Pazzani, M.: Derivative Dynamic Time Warping. First SIAM International Conference on Data Mining (2001)

[14]Kovalev D., Shanin I., Stupnikov S., Zakharov V.: Data Mining Methods and Techniques for Fault Detection and Predictive Maintenance in Housing and Utility Infrastructure. Engineering Technologies and Computer Science (2018)

[15]Köhler, Torsten, Lorenz: A comparison of denoising methods for one dimensional time series. Zentrum für Technomathematik (2005)

[16]Marron J. S., Tsybakov A. B.: Visual error criteria for qualitative smoothing. Journal of the American Statistical Association (1995)

[17]Merkel D.: Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux J., vol. 2014 (2014)

[18]Mobley K.: An Introduction to Predictive Maintenance — 2nd edition (2002)

[19]Moritz S., Sardá A., Bartz-Beielstein T., Zaefferer M., Stork J: Comparison of different Methods for Univariate Time Series Imputation in R. CoRR abs/1510.03924 (2015)

[20]Nayak S., Misra B., Behera H.: Impact of Data Normalization on Stock Index Forecasting. International Journal of Computer Information

Systems and Industrial Management Applications (2014)

[21]OpenTSDB 2.3 documentation | HTTP API. http://opentsdb.net/docs/build/html/api_http/put.html

[22]Phan T., Poisson Caillault E., Lefebvre A., Bigand A.: Dynamic time warping-based imputation for univariate time series data. Pattern Recognition Letters (2017)

[23]pySpark Package Documentation. http://spark.apache.org/docs/2.1.0/api/python/pyspark.html

[24]Sakoe, Chiba: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Transactions on Acoustics, Speech and Signal Processing (1978)

[25]Sigoure B.: OpenTSDB: The distributed, scalable time series database. OSCON, vol. 11 (2010)

[26]Song S., Zhang A., Wang J., Yu P.: SCREEN: Stream Data Cleaning under Speed Constraints. ACM SIGMOD international conference on management of data (2015)

[27]Spark Streaming Programming Guide. https://spark.apache.org/docs/latest/streaming-programming-guide.html

[28]Spark Streaming + Kafka Integration Guide. https://spark.apache.org/docs/2.2.0/streaming-kafka-integration.html

[29]White T.: Hadoop: The Definitive Guide. O'Reilly Media; Forth Edition (2012)

[30]Zhang A., Song S., Wang J.: Sequential Data Cleaning: A Statistical Approach. ACM SIGMOD international conference on management of data (2016)