ODIN TS: a tool for the black-box evaluation of time series analytics

Niccolò Zangrando[1], Rocio Nahime Torres[1], Federico Milani[1], and Piero Fraternali[1]

Politecnico di Milano, Piazza Leonardo Da Vinci, Milano, IT
{`niccolo.zangrando, rocionahime.torres, federico.milani, piero.fraternali`}@polimi.it

Abstract. The increasing availability of time series datasets enabled by the diffusion of IoT architectures and the progress in the analysis of temporal data fostered by Deep Learning methods are boosting the interest in anomaly detection and predictive maintenance applications. The analysis of performance for these tasks relies on standard metrics applied to the entire dataset. Such indicators provide a global performance assessment but might not help a deep understanding of the model weaknesses. A complementary diagnostic approach exploits error categorization and ad-hoc visualizations. In this paper we present ODIN, an open source diagnosis framework for time series analysis that lets developers compute performance metrics, disaggregated by different criteria, and visualize diagnosis reports. ODIN is agnostic to the training platform and can be extended with application- and domain-specific meta-annotations and metrics with almost no coding. We show ODIN at work through two time series analytics examples.

Keywords: time-series · anomaly detection · predictive maintenance · model evaluation · error diagnosis

1 Introduction

Time series datasets collect observations sampled at different times. Recording can be continuous, when data are collected continuously in a given interval, or discrete, when data are recorded at set time intervals [5]. Based on the number of observations at each timestamp, the time series can be univariate or multivariate. Univariate time series log values generated by a single sensor, whereas multivariate time series record signals from multiple sensors simultaneously. Time series are used to study time-varying phenomena in many fields: in economy [6] (e.g., stock price trends), in medicine [23] (e.g., the progress of health variables) or in the industry [22] (e.g. the status or energy consumption of a machine). Given a time series dataset, different tasks can be performed to predict a specific attribute or event at a given timestamp or assign a label to a particular observation. The most common tasks can be summarized as follows:

– **Classification**: assigning a class label to a time series [11]. An example of this task is the classification of the human heartbeat to detect deceases [18].
– **Forecasting**: predicting future event/s. An example is to predict the future energy consumption of an appliance based on historical data [1].
– **Anomaly Detection**: identifying deviations from normal behavior [9, 4, 24]. An example is the identification of anomalies in HVAC systems [3].
– **Predictive Maintenance**: predicting when a piece of equipment is likely to fail and deciding which maintenance activity to perform to obtain a good trade-off between maintenance frequency and cost [24]. This objective could be pursued with classification approaches (identify if the appliance will fail within n days) or regression ones (predict the Remaining Useful Life, RUL, of an appliance).

The different tasks are usually evaluated by means of standard metrics such as Mean Absolute Error (MAE) or Precision and Recall. While these metrics are useful global indicators of the model performances, they provide little insight into the weaknesses of the models. For example, predicting a false positive close to a real anomaly is a less severe error than predicting it at a very distant time. Furthermore, information collected but not used during the model training phase could help understand the model performances. For example, in an industrial application it could be interesting to analyze if the performances vary across appliance versions or install locations. Similar analysis could be performed on any other attribute not exploited for training but available at diagnosis time.

This paper introduces ODIN TS, the extension for anomaly detection and predictive maintenance of the ODIN machine learning diagnosis tool. ODIN is an open-source, Python-based, black-box framework for error diagnosis initially conceived for generic classification and computer vision tasks. ODIN TS adds the implementation of the most widely adopted metrics for the anomaly detection and predictive maintenance tasks and proposes new analyses for anomaly detection, such as false positive error categorization. ODIN TS also enables the inspection of the time series dataset and of the related predictions by means of a visualizer with different functionalities.

The contributions of the paper can be summarized as follows:

– We summarize the most widely used metrics for time-series analysis.
– We describe their implementation in ODIN TS, an extensible framework for time series analytics error diagnosis.
– We introduce the novel analysis and visualizations supported by ODIN TS and exemplify them in an anomaly detection and a predictive maintenance task.

The paper is organized as follows: Section 2 summarizes the most common metrics used for time-series analysis, Section 3 describes the proposed framework and its functionalities, Section 4 presents some examples of how the tool can be employed and finally Section 5 concludes and provides insight into the future work.

Table 1. Metrics and analysis found in the literature for Time Series based on the different tasks. The value "yes" is used to indicate the metric applies to the specific task, whereas "n/a" is used to indicate the contrary.

	Classification	**Forecasting**	**Anomaly Detection**	**Predictive Maintenance**	
				Classification	**Regression**
Accuracy[15]	yes	n/a	yes	yes	n/a
Precision[15]	yes	n/a	yes	yes	n/a
Recall[15]	yes	n/a	yes	yes	n/a
F1 Score[15]	yes	n/a	yes	yes	n/a
Miss Alarm Rate[2]	yes	n/a	yes	yes	n/a
False Alarm Rate[21]	yes	n/a	yes	yes	n/a
NAB Score[19]	n/a	n/a	yes	n/a	n/a
Mean Absolute Error (MAE)[14]	n/a	yes	n/a	n/a	yes
Mean Squared Error (MSE)[14]	n/a	yes	n/a	n/a	yes
Root Mean Squared Error (RMSE)[14]	n/a	yes	n/a	n/a	yes
Matthews Coefficient[8]	yes	n/a	yes	yes	n/a
Mean Absolute Percentage Error (MAPE)[14]	n/a	yes	n/a	n/a	yes
Precision-Recall Curve[20]	yes	n/a	yes	yes	n/a
ROC Curve[15]	yes	n/a	yes	yes	n/a
Gain & Lift Analysis[17]	yes	n/a	yes	yes	n/a
Residuals Analysis[26]	n/a	yes	n/a	n/a	yes
Coefficient of Variation[14]	n/a	yes	n/a	n/a	yes
Mean Absolute Ranged Relative Error (MARRE)[14]	n/a	yes	n/a	n/a	yes
Mean Absolute Scaled Error (MASE)[14]	n/a	yes	n/a	n/a	yes
Overall Percentage Error (OPE)[14]	n/a	yes	n/a	n/a	yes
Coefficient of Determination R2[14]	n/a	yes	n/a	n/a	yes
Rho-risk[14]	n/a	yes	n/a	n/a	yes
Root Mean Squared Log Error (RMSLE)[14]	n/a	yes	n/a	n/a	yes
Symmetric Mean Absolute Percentage Error (sMAPE)[14]	n/a	yes	n/a	n/a	yes

2 Related work

The evaluation of inference models applies standard metrics to compute performance indicators based on a comparison between the ground truth (what is expected) and the model predictions. Table 1 presents a by-task summary of the metrics and performance indicators found in the literature for time series analytics and provides a reference to the definition of each index. Different works have focused on the evaluation of time series tasks by proposing novel metrics and assessment procedures or by providing efficient implementations of the classical indicators. In [19] the authors present a new benchmarking metric, Numenta Anomaly Benchmark (NAB) score, which augments traditional indices by incorporating time explicitly so as to reward early detection and introduces the concept of anomaly window. The work in [12] illustrates the benefits of decomposing performance metrics based on the characteristics of the observations. As a use case a model to detect illicit use of computational resources is created and assessed. The evaluation considers how the performances of the predictor changes in servers with a specific attributes (low or high profile). Other contributions propose novel time series evaluation frameworks. The work [14] introduces Darts, a python framework for handling time series which implements some state-of-the-art machine learning methods and provides off-the-shelf a subset of the standard metrics reported in Table 1. Another example is the RELOAD tool proposed in

[28] to identify the most informative features of a dataset, run anomaly detection algorithms, and apply a set of evaluation metrics on the results. While both tools aim to support training and evaluation with standard metrics, ODIN TS extends the support to time series analytics with a black-box error diagnosis approach focused on the anomaly detection and predictive maintenance tasks. It enables error categorization, predictions decomposition and visualizations. The decomposition and visualization functionalities exploit meta-annotations of the data set, i.e., features not used during model training that can contribute to the interpretation of the model results.

3 The ODIN TS framework

The ODIN TS framework supports the development of predictive maintenance and anomaly detection tasks enabling designers to evaluate standard metrics on inputs and outputs grouped by meta-annotation values, perform error categorization, evaluate the confidence calibration error, and visualize a variety of diagnostic reports. ODIN TS also includes a Visualizer module for the inspection of the dataset and of the model predictions. ODIN TS is supported by the extension of classes in the Python-based ODIN framework, and publicly released[1].

3.1 Dataset input and output formats

ODIN TS supports the import of time series data, of ground truth (GT) annotations and the output of inference results. The artifacts should follow the guidelines common to most publicly available datasets (summarized in Table 2):

- **Anomaly Detection Task**
  - Time Series data: a CSV file with the column *timestamp* of the observation and one additional column for each *signal*.
  - Ground Truth: a JSON file containing a list of the *timestamp* in which the anomalies appear.
  - Predictions: a CSV file where the first column specifies the *timestamp* and the following column(s) the $confidence$ value, or the reconstructed/predicted *signal* values.
- **Predictive Maintenance Task**
  - Time Series data: a CSV with the columns *observation_id* for the unique identifier of the observation, *unit_id* for the machine or appliance identifier, and one additional column for each *signal*.
  - Ground Truth: it is embedded in the previous CSV file as an additional column (*label*) with a Boolean value denoting if the machine is going to fail within n timestamps (for classification) or with the RUL value (for regression).

[1] https://github.com/rnt-pmi/odin

Table 2. Dataset format for ODIN time series. The value "n/a" indicates that a field is not used.

	Anomaly Detection			Predictive Maintenance		
	Format	Row identifier	Signals/Values	Format	Row identifier	Signals/Values
Time Series	CSV	timestamp	a column per signal	CSV	observation_id, unit_id	a column per signal
Ground Truth	JSON	n/a	list of timestamps when the anomalies occur	embedded in TS CSV	n/a	label column (class) or RUL column (regr)
Properties	CSV	timestamp	a column per property	CSV	observation_id, unit_id	a column per property
Predictions	CSV	timestamp	confidence column or a column per signal	CSV	observation_id	confidence column (class) or RUL column (regr)

- Predictions: a CSV file (named *unit_id.csv*) for each machine or appliance mentioned in the ground truth. The file contains one column with the identifier of the observation and one column with the $confidence$ score (for classification) or with the RUL value (for regression).

In both cases if additional meta-properties are associated to the time series, these are input as a CSV where the first column is the identifier of the time series observation (*timestamp* or *observation_id*) and there is one column per meta-property.

3.2 Supported types of dataset analysis

ODIN TS supports the following types of analysis of the observations and of the ground truth annotations.

Distribution of classes. For classification, a plot displays the percentage of samples for each category.

Distribution of properties. A plot displays the percentage of the observations associated with each property value. For example, it visualizes if an observation is associated with a certain period of the day (morning, evening, night) or with a specific type of anomaly (point, contextual or collective).

Stationariety analysis. A stationary time series is one whose properties do not depend on the time at which the series is observed. The implementation of ODIN TS uses the Augmented Dickey-Fuller statistical test [7].

Seasonality, trend and residual decomposition. These analyses expose the repeating short-term cycles (seasonal) and the general movement over time (trend) of the series [16]. Residuals include everything not captured by the previous two types of decomposition. Decomposition can be realized with an additive model (addition of the decomposed values restores the original times series) or with a multiplicative one (the original series is obtained by multiplying the decomposed values).

Table 3. Types of analysis in ODIN TS for the different tasks. The value "n/a" specifies the type is not relevant for the specific task.

	Anomaly Detection	**Predictive Maintenance**	
		Classification	**Regression**
Summary report	**yes**	**yes**	**yes**
Performance per threshold value	**yes**	**yes**	n/a
Per property analysis	**yes**	**yes**	**yes**
FP anomaly categorization	**yes**	n/a	n/a
Error distance distribution	**yes**	n/a	n/a
RUL variation distribution	n/a	n/a	**yes**
Calibration analysis	**yes**	**yes**	n/a

3.3 Supported types of prediction analysis

ODIN TS implements all the metrics of Table 1. To the best of our knowledge, there is no other framework that offers all of them off-the-shelf. Based on the implemented metrics, ODIN TS implements multiple performance reports and types of prediction analysis, summarized in Table 3.

Summary report. A report that tabulates the results of all the metrics. The total shows both the micro- and macro-averaged values: the first computes the value without distinguishing the categories; the latter computes the metrics for each class and then performs an unweighted mean.

Performance per threshold value. The classification metrics of interest are computed and shown in a graph for each value of the confidence threshold.

Per property analysis. The values of the metrics of interest are decomposed by property value and contrasted with the average across all the property values. For example, the RUL value prediction or probability of failure in the next N timestamps could be distinguished per appliance brand or installation location.

FP anomaly categorization. The analysis of incorrectly predicted anomalies is supported, including their categorization into the following cases:

– *Affected*: an FP anomaly prediction is assigned to this category if its timestamp lies within an *anomaly window*. The anomaly window, introduced in [19], is an interval centered at the GT anomaly timestamp. The window extension (i.e.between, number of points) is a customizable parameter set by default to 10% of the data points divided by the number of anomalies.
– *Continuous*: this category contains FP anomalies that occur at contiguous timestamps outside the anomaly window.
– *Generic*: all the other FP anomalies.

FP error distance distribution. A distribution plot of the distance (measured as a number of timestamps) between a FP and the closest GT anomaly, color-coded with the FP anomaly category (affected, continuous or generic).

RUL variation distribution. Given that a machine or appliance degrades over time, the predicted RUL should decrease by 1 at each timestamp. In a perfectly consistent model the following formula should apply:

$$\hat{y}_t - \hat{y}_{t-1} = -1 \tag{1}$$

where $\hat{y}_t$ is the predicted RUL at time t and $\hat{y}_{t-1}$ is the predicted RUL at time t-1. This type of analysis plots the distribution of the differences of the predicted RUL between the current cycle and the previous one so as to assess the consistency of the model predictions.

Calibration analysis. It exploits the confidence histogram and the reliability diagram [10]. Both plots assign the confidence values to buckets (e.g., 0-0.1, 0.1-0.2, ..., 0.90-1) on the abscissa. The confidence histogram shows the percentage of positive predicted samples that fall into each confidence range. The reliability diagram indicates, for each confidence range, the average accuracy of the positive samples in that range. When a classifier is well-calibrated, its probability estimates can be interpreted as correctness likelihood, i.e., of all the samples that are predicted with a probability estimate of 0.6, around 60% should belong to the positive class [13]. ODIN reports the Expected Calibration Error (ECE) (Eq. 2) and the Maximum Calibration Error (MCE) (Eq. 3)

$$ECE = \sum_{m=1}^{M} \frac{B_m}{n} acc(B_m) - conf(B_m) \tag{2}$$

$$MCE = max_{m\epsilon(1..M)} |acc(B_m) - conf(B_m)| \tag{3}$$

where n is the number of samples in the data set, M is the number of buckets (each of size 1/M), and B_m denotes the set of indices of observations whose prediction confidence falls into the interval m.

3.4 Supported visualizations

ODIN TS allows one to visualize the dataset and the corresponding model predictions, if provided. The dataset visualization offers the following functionalities:

- Feature filter: one can choose which features to visualize of a multi-variate dataset.
- Aggregation: data can be aggregated by minute, hour, day, week, month or year and visualized at different granularity.
- Pagination: some datasets span a large interval. A pagination function with custom data points size and step can be used to browse the dataset.
- GT display toggle: the GT annotations can be shown or hidden. For anomaly detection, it could be a single point or an anomaly window. For predictive maintenance, the class labels or the RUL values.

If the predictions are available, the following functionalities can be used:

- Predictions visualization and model comparison: the predictions are visualized along with the GT. If multiple models are selected, their predictions are color-coded.
- FP errors visualization: the FP predictions are displayed and, in the case of FP anomalies, color-coded by their type.

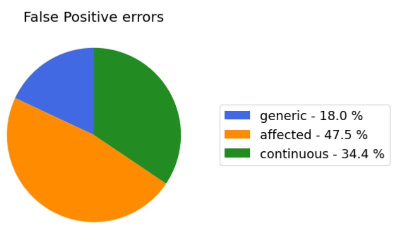

Fig. 1. False Positive distribution plot

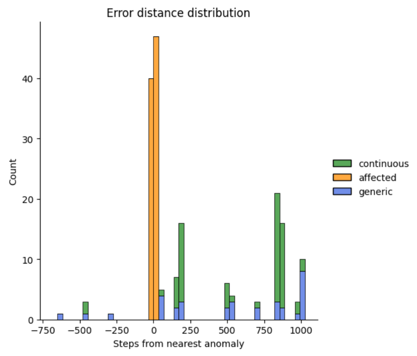

Fig. 2. False positive distribution of the distance of each FP from the closest anomaly.

4 ODIN TS in action

This section exemplifies ODIN TS at work on an anomaly detection and a predictive maintenance case. The first example from the NAB datasets [19] uses the *ambient temperature system failure* data which contains ≈ 5000 hourly temperature measurements in an office and features two anomalies. To detect the anomalies, an LTSM model was trained as in [4]. It comprises two LSTM modules with 4 hidden layers and 0.2 dropout rate. The training set includes the first 30% of the data (of which the 10% was used for validation) while the remaining 70% was been used for testing. The model was trained for 100 epochs, with a batch size of 32 and an input window length of 30. The scenario is relatively small given the few anomalies but is still useful to highlight some of the ODIN TS capabilities. Figure 1 shows the distribution of the FP error categories. For the computation of "affected" errors, an anomaly window of length 34 is used. Most FPs are within a short distance from the real anomalies ("affected", in orange), which suggests that the anomaly is perceived before its reported occurrence time and continues to be perceived shortly after. Also "continuous" FPs are more numerous than generic ones, which shows that the model tends to identify prolonged anomalies rather than instantaneous exceptions. Figure 2 shows the distance of each FP from the closest anomaly to confirm that the "affected" FPs are the closest to a GT anomaly. These errors are better appreciated in Figure 3 in which the Visualizer helps understand the findings of the analysis more intuitively.

To illustrate a case of predictive maintenance we use a NASA dataset about turbofan engine degradation [25]. The dataset comprises samples from 100 engines whose state is represented by 24 variables. The GT consists of the RUL at each inter-observation interval (called *cycle*). The authors provide the dataset in two splits: training (with 20631 observations among 100 machines) and testing (13096 observations among 100 machines). In the testing split the cycles for

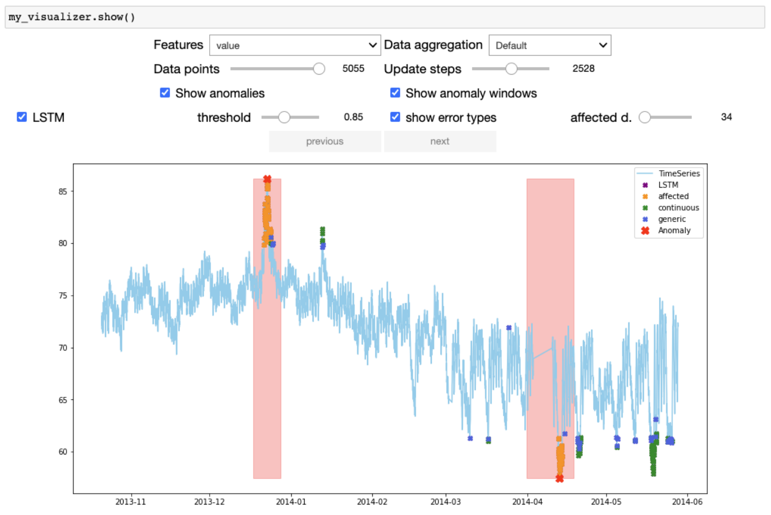
```
my_visualizer.show()
```
Features: value; Data aggregation: Default
Data points: 5055; Update steps: 2528
☑ Show anomalies; ☑ Show anomaly windows
☑ LSTM; threshold: 0.85; ☑ show error types; affected d.: 34

Fig. 3. The ODIN TS Visualizer showing the actual data and the model predictions color-coded by error type.

each machine range from 31 to 303. To predict the RUL at each cycle, a 2-layers LSTM was employed, with 128 and 64 hidden layers respectively and the same dropout rate of 0.3. The LSTM was trained for 10 epochs with a batch size of 150 and a window input length of 60 cycles.

The model has a RUL estimation error of 20 cycles (provided by the MAE) and an MAPE of $\approx$ 0.17, which denote good performance. Further analysis helps understand where the model can be improved. Figure 4 shows the residual analysis and enables a visual interpretation of the deviation of the predictions from the GT. Two plots report the predicted RUL on the X-axis. The Y-axis of the left plot reports the GT RUL value, while the Y-axis of the right plot reports the standardized difference between the prediction and the GT. Each

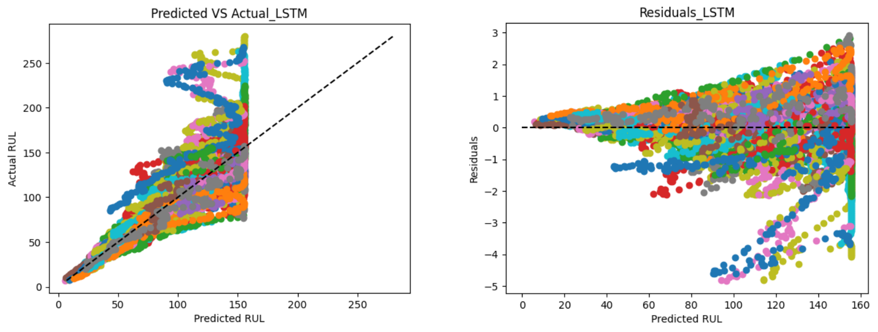

Fig. 4. Residuals analysis. The colors indicate the different engines.

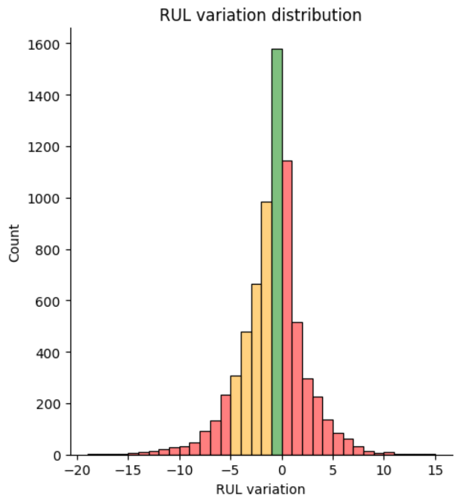

Fig. 5. RUL variation distribution: green bar represents a perfect variation of the RUL estimation, the yellow ones an acceptable variation and the red one an inconsistent prediction.

color represents a different engine. From the analysis, it can be seen that most errors occur when the component is still in good conditions (with a RUL value greater than 100), which highlights the inability of the model to predict the engine remaining life in the long term. In particular, the highest predicted RUL is 160, while the corresponding GT value is 260. This suggests that the model is not able to learn high RUL values properly. In scenarios where analysts are more interested in correctly predicting the RUL when the engine is close to a failure, it makes sense to set a maximum GT RUL value to reduce the relevance of large values during training [27].

Figure 5 illustrates the RUL variation distribution analysis, which shows that the model predictions are not very consistent. The model sometimes increases the estimated RUL by 10 cycles instead of decreasing it by 1. This finding can help improve the prediction; for example, if the variation among consecutive cycles is high, one might interpolate or average the RUL values of previous cycles to mitigate the noise.

5 Conclusions

This paper has described the addition of time series analysis functions into a black-box error diagnosis framework originally conceived for CV tasks. The novel version of ODIN includes an ODIN TS module which supports performance diagnosis for two analytics tasks on time series: anomaly detection and predictive maintenance. ODIN TS implements all the most widely adopted metrics for the addressed tasks and introduces new types of analysis for anomaly detection, such as FP error categorization. ODIN TS also enables the inspection of the dataset and of the predictions by means of a Visualizer with rich functionalities. ODIN TS is implemented in Python and released as an open-source project which developers can easily extend with their own metrics, reports and visualizations. To conclude we have illustrated the tool at work on two use cases, so as to give a glimpse of its utility. Future work will focus on extending the implemented metrics and on supporting more tasks (e.g., time series classification and forecasting). We also plan to extend the visualizer with novel functions and to integrate a time series annotator for enriching the GT or generating it from scratch (e.g., by annotating the anomalies in a dataset). Finally, we plan to create automatic property extractors (e.g., for assigning to each anomaly the proper type [9]).

6 Acknowledgments

This work is partially supported by the project "PRECEPT - A novel decentralized edge-enabled PREsCriptivE and ProacTive framework for increased energy efficiency and well-being in residential buildings" funded by the EU H2020 Programme, grant agreement no. 958284.

References

1. Ahmad, A.S., Hassan, M.Y., Abdullah, M.P., Rahman, H.A., Hussin, F., Abdullah, H., Saidur, R.: A review on applications of ann and svm for building electrical energy consumption forecasting. Renewable and Sustainable Energy Reviews **33**, 102–109 (2014)
2. An, N., Weber, S.: Impact of sample size on false alarm and missed detection rates in pca-based anomaly detection. In: 2017 51st Annual Conference on Information Sciences and Systems (CISS). pp. 1–6. IEEE (2017)
3. Beghi, A., Brignoli, R., Cecchinato, L., Menegazzo, G., Rampazzo, M., Simmini, F.: Data-driven fault detection and diagnosis for hvac water chillers. Control Engineering Practice **53**, 79–91 (2016)
4. Braei, M., Wagner, S.: Anomaly detection in univariate time-series: A survey on the state-of-the-art. arXiv preprint arXiv:2004.00433 (2020)
5. Brockwell, P.J., Davis, R.A.: Time series: theory and methods. Springer Science & Business Media (2009)
6. Centoni, M., Cubadda, G.: Modelling comovements of economic time series: a selective survey (2011)
7. Cheung, Y.W., Lai, K.S.: Lag order and critical values of the augmented dickey–fuller test. Journal of Business & Economic Statistics **13**(3), 277–280 (1995)
8. Chicco, D., Jurman, G.: The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. BMC genomics **21**(1), 1–13 (2020)
9. Choi, K., Yi, J., Park, C., Yoon, S.: Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. IEEE Access (2021)
10. DeGroot, M.H., Fienberg, S.E.: The comparison and evaluation of forecasters. Journal of the Royal Statistical Society: Series D (The Statistician) **32**(1-2), 12–22 (1983)

11. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data mining and knowledge discovery **33**(4), 917–963 (2019)
12. Gomes, G., Dias, L., Correia, M.: Cryingjackpot: Network flows and performance counters against cryptojacking. In: 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA). pp. 1–10. IEEE (2020)
13. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning. pp. 1321–1330. PMLR (2017)
14. Herzen, J., Lässig, F., Piazzetta, S.G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., et al.: Darts: User-friendly modern machine learning for time series. arXiv preprint arXiv:2110.03224 (2021)
15. Hossin, M., Sulaiman, M.N.: A review on evaluation metrics for data classification evaluations. International journal of data mining & knowledge management process **5**(2), 1 (2015)
16. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (2018)
17. Jaffery, T., Liu, S.X.: Measuring campaign performance by using cumulative gain and lift chart. In: SAS Global Forum. p. 196 (2009)
18. Kampouraki, A., Manis, G., Nikou, C.: Heartbeat time series classification with support vector machines. IEEE transactions on information technology in biomedicine **13**(4), 512–518 (2008)
19. Lavin, A., Ahmad, S.: Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In: 2015 IEEE 14th international conference on machine learning and applications (ICMLA). pp. 38–44. IEEE (2015)
20. Miao, J., Zhu, W.: Precision–recall curve (prc) classification trees. Evolutionary intelligence pp. 1–25 (2021)
21. Pokrywka, R.: Reducing false alarm rate in anomaly detection with layered filtering. In: International Conference on Computational Science. pp. 396–404. Springer (2008)
22. Qiao, H., Wang, T., Wang, P., Qiao, S., Zhang, L.: A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series. Sensors **18**(9), 2932 (2018)
23. Radhakrishnan, N., Gangadhar, B.: Estimating regularity in epileptic seizure time-series data. IEEE engineering in medicine and biology magazine **17**(3), 89–94 (1998)
24. Ran, Y., Zhou, X., Lin, P., Wen, Y., Deng, R.: A survey of predictive maintenance: Systems, purposes and approaches. arXiv preprint arXiv:1912.07383 (2019)
25. Saxena, A., Goebel, K.: Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository pp. 1551–3203 (2008)
26. Sheather, S.: A modern approach to regression with R. Springer Science & Business Media (2009)
27. Song, Y., Shi, G., Chen, L., Huang, X., Xia, T.: Remaining useful life prediction of turbofan engine using hybrid model based on autoencoder and bidirectional long short-term memory. Journal of Shanghai Jiaotong University (Science) **23**(1), 85–94 (2018)
28. Zoppi, T., Ceccarelli, A., Bondavalli, A.: Evaluation of anomaly detection algorithms made easy with reload. In: 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE). pp. 446–455. IEEE (2019)