

Systematic Review of Time-Series Machine Learning from a Networking Perspective

Liyan Qu, Johannes Späth*

**Chair of Network Architectures and Services*

School of Computation, Information and Technology, Technical University of Munich, Germany

Email: liyan.qu@tum.de, spaethj@net.in.tum.de

Abstract—Time-series machine learning (TSML) aims to handle data with temporal dependencies, which differs from traditional machine learning (ML) that assumes samples are independent and identically distributed (i.i.d.). TSML needs to consider dynamic characteristics such as trends, seasonality, and concept drift. Therefore, developing specific TSML tools is necessary. In recent years, plentiful TSML tools have emerged, but most studies focus on algorithm principles or single tasks, lacking systematic comparisons from the perspectives of tasks and tool characteristics. This paper takes a task-oriented approach to review and compare TSML tools covering major application scenarios, focusing on six tasks: imputation, feature extraction, forecasting, anomaly detection, change point detection, and classification. Based on this, the paper summarizes key characteristics to consider when selecting tools, such as time index awareness, backtest, multivariate support, scalability and output format. The review results can provide structured references for researchers without a machine learning background, helping them choose suitable TSML tools in practical engineering scenarios.

Index Terms—time-series, machine learning, networking, forecasting, anomaly detection

1. Introduction

Machine learning (ML) has achieved significant success in various fields such as image recognition, speech analysis, and financial forecasting [1]. In recent years, ML methods have also been applied to computer networking tasks, such as traffic forecasting, intrusion detection, and service classification [2]. Network-related data (e.g., traffic, latency, bandwidth utilization) naturally change continuously over time and are typical time-series data. However, time-series data often do not satisfy the independent and identically distributed (i.i.d.) assumption of traditional ML, as temporal dependencies, trends, seasonality, and concept drift exist between data samples [3]. Therefore, directly applying of traditional ML methods to time-series data is a significant challenge, and specific tools for TSML are required.

Recently, a large amount of research has been conducted on time-series machine learning (TSML) [4]. Existing reviews often focus on specific tasks [5], but systematic studies from the perspectives of tasks and tools remain limited. Especially for researchers without a machine learning background, there is still a lack of comprehensive review to guide practical tool selection. This

paper aims to fill this gap by systematically reviewing and comparing actively maintained open-source tools (such as tsfresh, sktime, Darts, and Kats) from the perspective of TSML tasks. Rather than delving into the internal algorithms or model architectures, this paper focus on analyzing the tools' functional characteristics and practical applicability. While Python-based libraries (such as tsfresh, sktime, Darts, and Kats) are the main focus, several representative tools in other programming languages are also briefly discussed. The selected tools are mostly actively maintained open-source libraries that cover common time-series tasks such as imputation, feature extraction, forecasting, anomaly detection, change point detection, and classification.

This paper starts from the Darts library. We searched Google Scholar and GitHub using keywords such as "time series machine learning", "time series forecasting", and "time series review", and then filtered out the mainstream tools that are still actively maintained. Based on their official documentation and functional descriptions, we categorized and organized them according to typical TSML tasks, with an aim to cover a sufficient number of representative tools for each task type.

The structure of this paper is as follows: the second section introduces the essential prerequisite knowledge; the third section systematically analyzes various tools and their characteristics according to task types; the fourth section provides summary and limitations of this paper.

2. Prerequisite knowledge

In this section, we introduce some concept that will be used in the following sections.

2.1. Supervised and Unsupervised Learning

In terms of what feedback can the learning model received, there are supervised learning, unsupervised learning [6]. Supervised learning models rely on data with class labels. Unsupervised learning does not rely on labels, it extracts patterns and structures from data spontaneously.

2.2. Online and Batch Learning

In terms of how data is fed into the learning model, there are online learning and batch learning [6]. In online learning, data is fed into the model one sample at a time, and the model updates itself incrementally with each new

sample. In batch learning, the model is trained on the entire dataset at once, and does not update itself until it is retrained with a new batch of data.

2.3. Features of TSML

The concepts below represent characteristics arising from the temporal nature of time-series data.

Peek into the future: using data information from future time points during model training or validation, which can lead to overly optimistic evaluation results and model overfitting [7].

Trend: a long-term sustained upward or downward direction in the time series, reflecting overall changes in the system over longer time scales [7].

Seasonality: periodic fluctuation patterns in the time series over time, such as daily, weekly, monthly, or annual cycles [7].

Concept drift: changes in data distribution or generation mechanisms over time, leading to model performance degradation on future data [8].

3. TSML Task Types and Tools

This section introduces common task types in TSML and their corresponding tools. The content is organized in a bottom-up sequence: basic mathematics and statistical tools, preprocessing tools, specific task tools, and comprehensive frameworks.

To provide clear guidance for selection, for each task type, this paper presents task definition, typical networking problems, key features that tools of this type should possess, and a comparison table of tools along with its explanation.

3.1. Foundational Libraries

Most TSML tools are based on similar underlying numerical and data processing libraries. NumPy provides matrix and vector operations [9], SciPy extends statistical and optimization methods [10], and pandas manages structured data and time indexing [11]. This section does not detail the specific dependencies of each tool, only indicates that, it is important to ensure the correct installation of relevant libraries.

3.2. Data Preprocessing

Raw time-series data often have problems such as redundancy, missing and noisy data, which adversely affect subsequent task processing. To improve performance and stability and reduce the overhead of downstream models, data preprocessing is usually required before implementing specific tasks, to reduce the burden on downstream algorithms.

3.2.1. Imputation. Imputation [12] refers to estimating and filling in missing values in time-series data by leveraging existing observed data points. Its goal is to reconstruct data integrity and avoid model deviation or performance reduction due to missing data. For example, as discussed

in [13], in network traffic monitoring, packet loss during sampling or transmission latency often leads to observation data missing, which needs to be reconstructed through imputation to ensure the accuracy of subsequent processes such as, anomaly detection or capacity planning.

When selecting imputation tools, the following five aspects should be considered:

- **Time-Index-Aware:** Whether the tool can recognize and utilize timestamps, and adjust imputation logic according to time intervals.
- **No-Lookahead:** Whether the tool prevents using future observations during imputation, thus avoiding data leakage or causal violations.
- **Multivariate Support:** Whether the tool can jointly impute multiple correlated variables (columns) instead of processing each variable independently.
- **Pipeline-Compatible:** Whether the tool provides standardized interfaces or APIs that allow seamless integration into machine-learning pipelines.
- **Probabilistic Output:** Whether the tool can provide probabilistic estimates, such as confidence intervals, quantiles, or sample distributions, in addition to single point estimates.

This section introduces six representative imputation libraries commonly used in TSML: pandas-interpolate() [14], SciPy-interpolate [15], sklearn-SimpleImputer [16], sklearn-KNNImputer [17], sklearn-IterativeImputer [18], and statsmodels-SARIMAX [19]. Table 1 compares these imputation tools in terms of the above features.

All comparison tables in this chapter use the following notation: "yes" indicates that the tool directly supports the feature, "partial" indicates that it is partially implemented only under specific conditions or in a limited way, "user implemented" indicates that it can be implemented through user-defined code, and "no" indicates that it is not supported at all.

3.2.2. Feature Extraction. Feature extraction [20], also known as representation learning, refers to the process of transforming raw or high-dimensional data into structured numerical features. Most ML models cannot directly handle raw unstructured data. Therefore, emphasizing important information and denoising through feature extraction, enables the data to be directly used by models. Feature extraction can reduce training costs, improve information relevance, enhance interpretability (although sometimes it may reduce it), and is often accompanied by dimensionality reduction.

Moreover, as we mentioned before, unlike traditional static ML, TSML also needs to capture temporal dependencies, trends, seasonality, and volatility. Therefore, when using non-deep learning models, specialized time-series feature extraction tools are required.

For instance, after missing traffic data are imputed, feature extraction is often applied to transform raw measurements into statistical indicators, such as packet inter-arrival variance, flow duration, or burstiness metrics, that help downstream anomaly-detection models identify network congestion or attacks.

When selecting feature-extraction tools, the following six aspects should be considered:

- **Programming Language:** The programming

TABLE 1: Comparison of Time-Series Imputation Tools

Tool	Time-Index-Aware	No-Lookahead	Multivariate Support	Pipeline-Compatible	Probabilistic Output
pandas — interpolate	yes	partial	no	partial	no
SciPy — interpolate	yes	no	no	no	no
sklearn — SimpleImputer	no	partial	no	yes	no
sklearn — KNNImputer	no	partial	yes	yes	no
sklearn — IterativeImputer	no	partial	yes	yes	user implemented
statsmodels — SARIMAX	yes	partial	partial	user implemented	yes

environments in which the tool is implemented.

- **Built-in Imputation:** Whether the tool provides built-in imputation functionality.
- **Multivariate Support:** Whether the tool can extract features from multiple variables (columns) at once.
- **Number of Features:** The number of features that the tool can extract; Catch22 can only extract a fixed set of 22 features.
- **Auto-Selection:** Whether the tool has mechanisms for automatically selecting features, rather than relying on user manual configuration.
- **Last Maintenance:** The date of the latest update to the tool, used to assess whether the project is still maintained and active.

This section introduces five representative feature extraction libraries commonly used in TSML: hctsa [21], Catch22 [22], tsfeatures [23], tsfresh [24], and TSFEL [25]. Table 2 compares these feature extraction tools in terms of the above features.

3.3. Specific Task Tools

According to different task goals, this section introduces four representative task types in TSML: forecasting, anomaly detection, segmentation/change-point detection, and classification.

3.3.1. Forecasting. Forecasting [26] is the most common and fundamental task type in TSML. Its goal is to utilize existing time-series data to build models, by capturing trends, seasonality, and random fluctuations, to predict explicit data values at future time points. Forecasting tasks require precise and stable results, to support decision-making and resource allocation.

Unlike traditional static ML forecasting, TSML forecasting emphasizes temporal continuity and causal constraints: training and testing must strictly follow the time order, and avoid the problem of "looking into the future". Moreover, modern TSML forecasting tools typically build a continuous loop: "forecast -> new data arrival -> monitoring -> model update", which is called online learning, rather than one-time training and output, known as batch learning.

As discussed in [27], link bandwidth or traffic variations forecasting is one of the typical applications in computer networks. Accurate traffic forecasting can help network systems to allocate resources and balance loads in advance, and to avoid congestion and performance degradation. Besides, network latency and packet-loss rate forecasting, data-center energy consumption forecasting, and server workload and task volume forecasting are also very practical applications [28].

When selecting forecasting tools, the following five aspects should be considered:

- **Backtest:** Whether the tool has built-in back test functionality, to prevent "looking into the future".
- **Online Update:** Whether the tool can continuously update models when new data arrive, to adapt to concept drift.
- **Output Format:** The types of forecast results the tool can generate—such as point estimates, prediction intervals, quantile values, sampled forecasts, or full predictive distributions. Richer outputs enable better uncertainty quantification.
- **Covariates Support:** Whether the tool allows incorporating external variables (such as temperature, traffic, or network load) to improve forecasting accuracy.
- **Dataset Scale:** The scale of datasets that the tool can efficiently handle, in terms of number of time series and length of each series. According to Hyndman [7], it can be classified as: Small: dozens to hundreds of series, each with less than 1000 time steps; Medium: hundreds to thousands of series, each with 1000-10000 time points; Large: tens of thousands of series, or each series exceeding 10000 time points.

This section introduces six representative forecasting libraries commonly used in TSML: Prophet [29], statsmodels-SARIMAX [30], StatsForecast [31], Darts [32], PyTorch-Forecasting [33], GluonTS [34], and ML-Forecast [35]. Table 3 compares these forecasting tools in terms of the above features.

3.3.2. Anomaly Detection. Anomaly detection [36] refers to the process of identifying data points or segments in a time series that significantly deviate from normal patterns. Its goal is to detect anomalous behaviors or potential system errors in time to prevent losses or performance degradation. Unlike static anomaly detection in traditional ML, TSML anomaly detection must consider temporal dependencies, periodic patterns, and inter-variable correlations.

As discussed in [37], distributed denial-of-service (DDoS) attack detection is a typical application scenario of anomaly detection in computer networks. This technique needs to identify sudden traffic surges or abnormal connection patterns in real time, to ensure network reliability. Intrusion detection, traffic matrix anomaly detection, and IoT network anomaly detection are also important applications.

When selecting anomaly detection tools, the following six aspects should be considered:

- **Backtest, Online Test, Multivariate Support:** Same as before.
- **Residual:** Whether the tool can remove trend and sea-

TABLE 2: Comparison of Time-Series Feature Extraction Tools

Tool Name	Programming Language	Built-in Imputation	Multivariable Support	Number of Features	Auto-Selection	Last Maintenance
hctsa	Matlab	no	no	7,700+	yes	2024-11-25
Catch22	C, Python, R Matlab, Julia	no	no	22	fixed 22	2022-06-21
tsfeatures	R	no	no	23+	no	2023-08-28
tsfresh	Python	no	yes	1200+	yes	2025-08-30
TSFEL	Python	yes	yes	65+	no (select by config)	2024-09-12

TABLE 3: Comparison of Time-Series Forecasting Tools

Name	Backtest	Online Update	Output Format	Covariates Support	Dataset Scale
Prophet	built-in	yes	point /intervals	yes	small–medium
statsmodels (SARIMAX / ETS)	user-implemented	yes	point /intervals	yes	small–medium
StatsForecast	built-in	yes	point /intervals	yes (limited)	medium–large
Darts	built-in	yes	point /quantiles	yes	small–medium
PyTorch Forecasting	partial	yes	intervals /quantiles	yes	medium–large
GluonTS	built-in	yes	samples /distributions	yes	medium–large
MLForecast	partial	partial	point /quantiles	yes	large

sonal components through forecasting models, and then detect anomalies in the residuals.

• **Anomaly Types:** The types of anomalies that the tool can identify, such as: Point Anomalies: Sudden spikes or drops at individual time points. Contextual Anomalies: Values that are normal in the overall trend, but anomalous under specific contexts or time conditions, e.g., high traffic at night. Subsequence Anomalies: Short sequences exhibiting anomalous patterns within a continuous time window, e.g., abnormal fluctuations within a cycle. Change-Point Anomalies: Persistent shifts in the baseline or statistical properties of the time series, e.g., significant changes in mean or variance. Multivariate Anomalies: Anomalous behaviors exhibited jointly across multiple time points or feature dimensions, which may appear normal when observed individually, but deviate from regular patterns overall.

This section introduces six representative time-series anomaly detection libraries: Merlion [38], ADTK [39], stumpy(Matrix Profile) [40], ruptures [41], Darts [32], and GluonTS [34]. Table 4 compares these anomaly detection tools in terms of the above features.

3.3.3. Change Point Detection and Segmentation Tools.

Change point detection (CPD) [42] aims to identify time-stamps in a time series where its statistical properties or structural patterns undergo significant and persistent changes, while segmentation divides the series into segments with similar behaviors and stable features based on these changes. Both tasks are usually implemented by the same tools, so they are introduced together in this paper. In TSML, this task is typically implemented in an unsupervised learning manner, without requiring predefined labels to detect changes. This process helps capture system status transitions, understand pattern evolution, and detect potential anomalies.

For example, as discussed in [43], online CPD on time series of latency and throughput in network measurements can be used to monitor real-time network service quality.

Besides, CPD can also be used for network traffic structure change detection, network device or link status switching detection, network topology evolution/routing change detection, and network quality-of-service (QoS) baseline drift detection [44].

When selecting CPD or segmentation tools, the following five aspects should be considered:

- **Backtest, Online Test, Multivariate Support:** Same as before.
- **Unsupervised Types:** Whether the model support unsupervised training.
- **Confidence Quantification:** The degree of confidence quantification for each candidate change point result. Confidence levels include: high, medium, and low. High: Outputs significant quantities (such as p-values, confidence intervals, posterior probabilities), or provides calibrated probability metrics. Medium: Outputs probabilistic scores with built-in thresholds or adaptive calibration tools. Low: Only outputs raw scores or cost values without statistical calibration. This section introduces six representative CPD libraries commonly used in TSML: ruptures [41], changepoint(R) [45], stumpy(Matrix Profile) [40], ADTK [39], and Kats [46].

Table 5 compares these CPD tools in terms of the above features.

3.3.4. Classification. Time-series classification [47] aims to categorize time-series samples into predefined classes based on their overall patterns or features. Unlike traditional ML settings where "each sample corresponds to features at a single time point", in TSML, each sample consists of multiple time points, and models need to capture temporal dependencies, trends, and shapes simultaneously, to identify behavior types or system states.

As discussed in [48], in the field of network security, time-series classification is often used for malicious network traffic detection. Models analyze time-series features of network packets or connections (such as packet size distribution, inter-arrival times, flow duration) to learn

TABLE 4: Comparison of Time-Series Anomaly Detection Tools

Name	Backtest	Online Update	Residual	Multivariable Support	Anomaly Types
Merlion	yes	yes	yes	yes	Point, Contextual, Collective
ADTK	partial	partial	partial	partial	Point, Contextual
stumpy (Matrix Profile)	user-implemented	partial	no	yes	Point, Subseq
ruptures	user-implemented	no	no	partial	Change
Darts/ GluonTS	yes	partial	yes	yes	Point, Contextual, Collective

TABLE 5: Comparison of Time-Series CPD and Segmentation Tools

Name	Backtest	Online Update	Multivariate Support	Unsupervised Types	Confidence Quantification
ruptures	partial	no	yes	yes	low
changepoint (R)	partial	no	user implemented	yes	high
stumpy (Matrix Profile)	partial	partial	user implemented	yes	low
ADTK	partial	partial	partial	yes	medium
Kats	yes	partial	yes	yes	high

temporal patterns of different traffic types, thereby automatically distinguishing normal communication from malicious activities. Besides, there are applications such as encrypted traffic classification, protocol identification, and user behavior modeling [49].

When selecting TSC tools, the following five aspects should be considered:

- **Backtest, Online Test, Multivariate Support:** Same as before.
- **Variable-Length Handling:** Whether it can handle sequences with inconsistent time steps or varying lengths among samples.
- **Interpretability:** Whether the tool provides feature-importance analysis, visualization, or rule-based outputs.

This section introduces five representative TSC libraries commonly used in TSML: sktime [50], tsai [51], tslearn [52], pyts [53], and tsfresh [24].

Table 6 compares these TSC tools in terms of the above features.

3.4. Comprehensive Frameworks

At the same time, several comprehensive frameworks can handle multiple tasks: statsmodels supports imputation and forecasting with classical models (ARIMA, SARIMAX, ETS). tsfresh focuses on feature extraction and imputation for downstream ML tasks. Darts covers forecasting and anomaly detection using both statistical and deep models. GluonTS is designed for forecasting and anomaly detection with probabilistic outputs. ruptures is specialized in change-point and anomaly detection. stumpy detects anomalies and change points via the Matrix Profile. ADTK is dedicated to anomaly and change-point detection.

4. Conclusion and Limitations

This paper has systematically presented the essential prerequisite knowledge for time-series machine learning, the core task types, and task-oriented tool comparisons. Due to space limitations, all tables are presented in the appendix. The goal is to provide a clear path to quickly

select appropriate tools for specific problems and to clarify the practical characteristics of each framework.

There are still some limitations: for each task type, only about five representative tools are included to cover approximately 90% of common scenarios; unconventional or niche applications are not discussed. Detailed algorithms, internal model architectures, and parameter settings of each library are beyond the scope of this paper.

Future work may expand this study by incorporating quantitative benchmarking, broader tool coverage, and unified evaluation criteria. Overall, this review serves as a structured reference for researchers and practitioners to navigate and select TSML tools effectively.

References

- [1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, 2015.
- [2] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, 2018.
- [3] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data mining and knowledge discovery*, 2017.
- [4] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," in *2019 International joint conference on neural networks (IJCNN)*. IEEE, 2019.
- [5] M. Rußwurm and M. Körner, "Self-attention for raw optical satellite time series classification," *ISPRS journal of photogrammetry and remote sensing*, 2020.
- [6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, 2016.
- [7] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.
- [8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, 2014.
- [9] C. R. Harris, K. J. Millman, S. J. van der Walt *et al.*, "Array programming with NumPy," 2020.
- [10] P. Virtanen, R. Gommers, T. E. Oliphant *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python."

TABLE 6: Comparison of Time-Series Classification Tools

Name	Backtest	Online	Multivariate	variable-length	Interpretability
sktime	partial	user implemented	yes	partial	yes
tsai	user implemented	user implemented	yes	user implemented	no
tslearn	user implemented	user implemented	partial	yes	partial
pyts	user implemented	user implemented	partial	partial	yes
tsfresh	user implemented	user implemented	yes	yes	yes

- [11] W. McKinney, “Data structures for statistical computing in python,” in *Proceedings of the 9th Python in Science Conference*, 2010.
- [12] Y. Xu and R. Goodacre, “On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning,” *Journal of analysis and testing*, 2018.
- [13] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, “An information-theoretic approach to traffic matrix estimation,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 301–312.
- [14] “pandas.DataFrame.interpolate,” pandas official documentation, accessed: 2025-09-28. [Online]. Available: <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html>
- [15] “scipy.interpolate,” SciPy official documentation, accessed: 2025-09-28. [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/interpolate.html>
- [16] “sklearn.impute.SimpleImputer,” scikit-learn official documentation, accessed: 2025-09-28. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html>
- [17] “sklearn.impute.KNNImputer,” scikit-learn official documentation, accessed: 2025-09-28. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html>
- [18] “sklearn.impute.IterativeImputer,” scikit-learn official documentation, accessed: 2025-09-28. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html>
- [19] “statsmodels.tsa.statespace.SARIMAX,” statsmodels official documentation, accessed: 2025-09-28. [Online]. Available: <https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html>
- [20] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*, 2008.
- [21] B. D. Fulcher and N. S. Jones, “hctsa: A computational framework for automated time-series phenotyping using massive feature extraction,” *Cell systems*, vol. 5, no. 5, pp. 527–531, 2017.
- [22] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, “catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis,” *Data mining and knowledge discovery*, 2019.
- [23] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2006.
- [24] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, “Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package),” *Neurocomputing*, vol. 307, pp. 72–77, 2018.
- [25] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, “Tsfel: Time series feature extraction library,” *SoftwareX*, vol. 11, p. 100456, 2020.
- [26] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne, “Machine learning strategies for time series forecasting,” in *Business Intelligence*, 2013.
- [27] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, “An information-theoretic approach to traffic matrix estimation,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003.
- [28] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, “Transformers in time series: A survey,” *arXiv preprint arXiv:2202.07125*, 2022.
- [29] “Prophet,” accessed: 2025-09-28. [Online]. Available: <https://facebook.github.io/prophet/>
- [30] “statsmodels,” accessed: 2025-09-28. [Online]. Available: <https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html>
- [31] “StatsForecast,” accessed: 2025-09-28. [Online]. Available: <https://nixtlaverse.nixtla.io/statsforecast>
- [32] “Darts,” accessed: 2025-09-28. [Online]. Available: <https://unit8co.github.io/darts/>
- [33] “PyTorch Forecasting,” accessed: 2025-09-28. [Online]. Available: <https://pytorch-forecasting.readthedocs.io/>
- [34] “GluonTS,” accessed: 2025-09-28. [Online]. Available: <https://ts.gluon.ai/>
- [35] “MLForecast,” accessed: 2025-09-28. [Online]. Available: <https://nixtlaverse.nixtla.io/mlforecast>
- [36] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, 2009.
- [37] K. Garg and R. Chawla, “Detection of ddos attacks using data mining,” *International Journal of Computing and Business Research (IJCBR)*, vol. 2, no. 1, pp. 2226–6166, 2011.
- [38] “Merlion,” accessed: 2025-09-28. [Online]. Available: <https://salesforce.github.io/Merlion/>
- [39] “Anomaly Detection ToolKit (ADTK) documentation,” accessed: 2025-09-28. [Online]. Available: <https://adtk.readthedocs.io/>
- [40] “STUMPY (Matrix Profile),” accessed: 2025-09-28. [Online]. Available: <https://stumpy.readthedocs.io/>
- [41] “ruptures,” ruptures documentation (change point detection), accessed: 2025-09-28. [Online]. Available: <https://centre-borelli.github.io/ruptures-docs/>
- [42] S. Aminikhanghahi and D. J. Cook, “A survey of methods for time series change point detection,” *Knowledge and information systems*, 2017.
- [43] C. Lévy-Leduc and F. Roueff, “Detection and localization of change-points in high-dimensional network traffic data,” *The Annals of Applied Statistics*, pp. 637–662, 2009.
- [44] S. Aminikhanghahi and D. J. Cook, “A survey of methods for time series change point detection,” *Knowledge and information systems*, 2017.
- [45] “R package ‘changepoint’ (CRAN),” accessed: 2025-09-28. [Online]. Available: <https://cran.r-project.org/web/packages/changepoint/index.html>
- [46] “Kats,” accessed: 2025-09-28. [Online]. Available: <https://facebookresearch.github.io/Kats/>
- [47] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” *Data mining and knowledge discovery*, 2019.
- [48] A. Shiravi, H. Shiravi, M. Tavallaei, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *computers & security*, 2012.
- [49] G. Aceto, D. Ciunzo, A. Montieri, and A. Pescapé, “Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges,” *IEEE transactions on network and service management*, 2019.
- [50] “sktime,” accessed: 2025-09-28. [Online]. Available: <https://www.sktime.org/en/stable/>
- [51] “tsai (timeseriesAI),” accessed: 2025-09-28. [Online]. Available: <https://timeseriesai.github.io/tsai/>
- [52] “tslearn,” accessed: 2025-09-28. [Online]. Available: <https://tslearn.readthedocs.io/>
- [53] “pyts,” accessed: 2025-09-28. [Online]. Available: <https://pyts.net/>