A Framework for the Evaluation of Clinical Time Series Models

Michael Gao
Department of Biostatistics and Bioinformatics
Duke University
Durham, NC
michael.gao@duke.edu

Jiayu Yao
SEAS
Harvard University
Cambridge, MA
jiy328@g.harvard.edu

Ricardo Henao
Department of Biostatistics and Bioinformatics
Duke University
Durham, NC
ricardo.henao@duke.edu

Abstract

Early detection of critical events is one of the mainstays of clinical time series prediction tasks. As data from electronic health records become larger in volume and availability increases, models that can predict critical events before they occur and inform clinical decision making have the potential to transform aspects of clinical care. There has been a recent surge in literature looking at early detection in the context of clinical time series. However, methods used to evaluate clinical time series models in which multiple predictions per time series are made often do not adequately measure the utility of the models in the clinical setting. Classical metrics such as the Area Under the Receiver Operating Characteristic (AUROC) and the Area Under the Precision Recall Curve (AUPRC) fail to fully capture the true, real-world performance of these models. In this work, we i) propose a method to evaluate early prediction models in a way that is consistent with their application in the clinical setting, and ii) provide a fast, open-source, and native cross-platform implementation.

1 Introduction

Models for early detection of several endpoints in the inpatient setting such as sepsis (Futoma et al., 2017; Henry et al., 2015), mortality (Kim et al., 2019), and acute kidney injury (Tomašev et al., 2019) have been developed using data from multivariate time series obtained from Electronic Health Records. Methods for reporting metrics of these models vary with respect to the unit of analysis. Though ROC and Precision-Recall analyses are reported in most cases, some authors report these in which each episode (*e.g.,* patient trajectory) is treated as a single unit (Kim et al., 2019), whereas in others metrics are also reported at the prediction (unit of time) level (Tomašev et al., 2019; Hyland et al., 2020). Some authors differentiate between episode-level and prediction-level metrics. In particular, Tomašev et al. (2019) reported per-step (prediction-level) precision-recall scores as well as episode-level metrics. (Hyland et al., 2020) evaluated precision at the prediction-level and recall (sensitivity) at the episode-level. In addition, due to the large number of predictions, Hyland et al. (2020) also introduce *snoozing* in order to account for the frequency of predictions and adjust for clinical workup time.

Workshop on Learning from Time Series for Health, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

There are several approaches for reporting metrics specifically for time series. Besides traditional metrics for binary classification such as AUROC and AUPRC, Lavin and Ahmad (2015) describe scoring functions which weight different components of the confusion matrix and utilize windows surrounding the event of interest to compute utility scores. Handler et al. (2022) also allow for problems-specific assignment of utility to confusion matrix components and additionally incorporate snoozing in order to reduce alert fatigue. Rather than looking at a single time point as the event of interest, Tatbul et al. (2018) extend Precision Recall analysis to working with event *ranges*. As an explicit trade-off between prediction-level precision and episode-level sensitivity, Garg et al. (2022) introduce a variant of the F-score that accounts for these differences. In this work, we show how specific choices of metrics to report translate to questions regarding the clinical utility of early detection models, make recommendations on metrics to report, and provide a fast and open source implementation of software to facilitate reporting of these metrics.

2 Early Detection for Clinical Time Series

In evaluating the classification performance of early detection models for clinical time series, we argue that there are two main questions of interest to users of a tool developed to detect events early in a clinical setting. These are:

- What proportion of events are detected sufficiently early by the model? Assuming "sufficiently early" has been determined in advance.
- What is the ratio of true alerts to false alerts generated by the model?

These questions are not always addressed by a traditional evaluation of early detection models. In particular, it is important to include both **episode-level** and **prediction-level** metrics in order to evaluate models in alignment with their clinical utility. Here, episode-level metrics refer to metrics that are computed over the entire time series for a given patient or event whereas prediction-level metrics refer to metrics that treat each prediction-label pair as its own unit.

2.1 Problem Definition

This work considers the early prediction task in which *multiple* predictions are made for a single time series, *e.g.*, patient trajectory. In this context, we will refer to each time series as an episode, allowing for flexibility in cases where they may be more than a single episode per patient trajectory. Let X_{ij} be a vector of patient covariates for patient i at time t_{ij}. For each patient indexed by $i \in \{1, 2, 3, ..., n\}$, we have a set of times t_{ij} where $j \in \{1, ..., m_i\}$. In addition, we let δ be a fixed detection window in which we are interested in predicting an outcome, and T_i be the time that patient i has the outcome of interest. If the patient does not have the outcome of interest, we set $T_i \to \infty$ (in practice this can be any sufficiently large number such that no positive predictions for time series where the event does not occur). Then, the corresponding labels $y_{ij} \in \{0, 1\}$ are given by the indicator function $\mathbb{1}(\{t_{ij} \in [T_i - \delta, T_i)\})$, *i.e.*, the label is 1 if the prediction time for patient i at time t_{ij} is within the prediction window and 0 otherwise. In this context, it makes sense to restrict $t_{ij} < T_i \, , \forall i, j$. The goal of early detection is to learn a model $h_\theta(\cdot)$ parameterized by θ such that $h_\theta(X_{ij})$ predicts y_{ij}, often through minimization of some criterion function $L(h_\theta(X), y)$.

2.2 Confusion Matrix Metrics

A natural construction of the components of the confusion matrix for early detection problems is to simply treat each prediction-label pair $\{h_\theta(X_{ij}), y_{ij}\}$ as independent units and categorize the prediction as either a True Positive (TP), False Positive (FP), True Negative (TN), or False Negative (FN) depending on some thresholded value (operational point) for $h_\theta(X_{ij})$. This is the standard treatment of a binary classification problem. For a given threshold z, we call a prediction 1 (positive) if $h_\theta(X_{ij}) \geq z$ and 0 (negative) otherwise. Under this setup, we can see that the confusion matrix values are given as follows: A **True Positive (TP)** refers to any time point *within* the detection window at which a positive prediction is made. A **False Positive (FP)** refers to any time point *outside* the detection window at which a positive prediction is made. A **True Negative (TN)** refers to any time point *outside* the detection window at which a negative prediction is made. A **False Negative (FN)** refers to any time point *inside* the detection window at which a negative prediction is made.

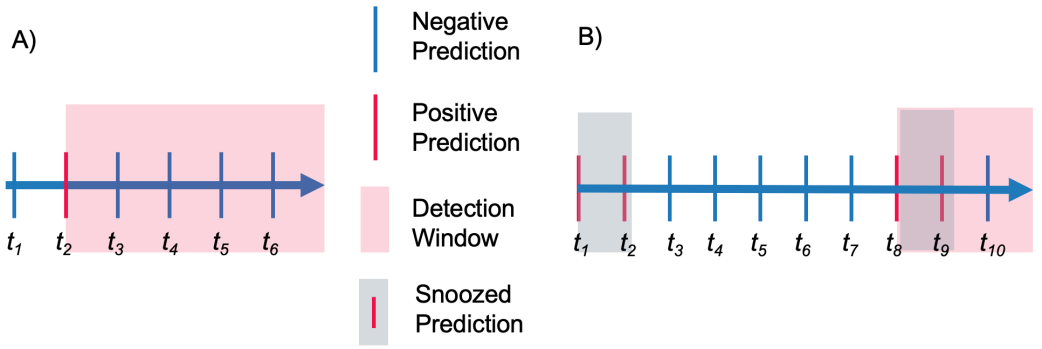

Figure 1: A) Example of a patient episode. In this example, one positive prediction (t_2) is made out of 5 total predictions (vertical lines) within the detection window. If we treat sensitivity at the prediction-level, the calculated sensitivity would be 20%. If we treat this at the episode level, then this represents one true positive, since a positive prediction was raised during the specified detection window. B) Illustration of snoozing. In this example, the gray shaded areas represent snoozed time points, so positive predictions t_2 and t_9 would be snoozed in the analysis.

These in turn lead to well-defined concepts such as the sensitivity or recall ($\frac{\#TP}{\#TP+\#FN}$), precision or positive predictive value ($\frac{\#TP}{\#TP+\#FP}$), and false positive rate ($\frac{\#FP}{\#FP+\#TN}$). Two primary methods of evaluation are to examine the ROC curve (trade-off between sensitivity and false positive rate), and the Precision-Recall curve (trade-off between precision and recall).

2.3 Sensitivity and Specificity in Clinical Medicine

The concept of sensitivity and specificity in medicine are often associated with diagnostic tests. Under this definition, the sensitivity is defined as the number of "Diseased persons with positive test results" divided by the "[Total Number of] All Diseased Persons" (Glaros and Kline, 1988), which – in the early prediction context – maps to the question "What proportion of patients who go on to experience an event of interest are detected by the model?". Under the specification of the confusion matrix values in Section 2.2, we can see that sensitivity instead answers the question "Of all of the time points (t_{ij}) which are within the detection window $[T_i - \delta, T_i)$, what proportion of them resulted in positive predictions?". As an illustration of the difference of these two questions, consider the following example. Suppose that a given patient has k predictions within their detection window, one of which had a positive prediction and the rest which were negative. Then according to the classical setup, the sensitivity for this patient would be only $1/k$, despite the fact that a positive prediction was made in the detection window. This is further illustrated in Figure 1A.

2.4 Episode-based Sensitivity and Specificity

Instead of the confusion matrix specification outlined in 2.2, which corresponds to a *prediction-level* metric, we redefine the confusion matrix specification to focus on detection as an *episode-level* metric in order to answer questions such as "what percentage of patients who experienced an event were captured early?" We define the episode-level confusion matrix as follows:

- TP: Any episode (*e.g.*, patient trajectory) which has at least one positive prediction within the detection window.
- FP: Any episode in which the event does not occur, but *does have* at least one positive prediction.
- TN: Any episode which has no positive predictions and the event does not occur.
- FN: Any episode in which the event *does* occur, but has no positive predictions.

Under this specification, the episode-level sensitivity or recall is defined as the proportion of episodes in which the event of interest occurs that has at least one positive prediction within the specified detection window.

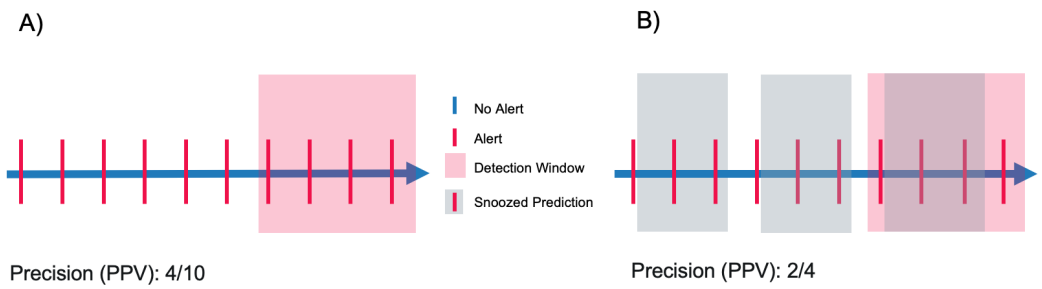

Figure 2: A) Under the presence of snoozing, the precision of this alert system is 4 true positives out of 10 total alerts. B) Under the presence of a fixed snoozing window size, the precision of the alert system is 2 true positives out of 4 total alerts. A change in precision often occurs under the presence of snoozing and is sensitive to the snoozing window size, the timing/frequency of alerts, and a variety of other factors.

2.5 Balancing True Positives and False Positives

The second question of interest as stated in 2 is related to the ratio of true alerts to false alerts generated by the model. This lends itself to the prediction-level confusion matrix definitions, where each prediction is treated as the unit of analysis. This makes intuitive sense, as a model which is trained to detect an event early may generate more than one alert and one may wish to assess the ratio of true positive to false positives in the context of individual predictions rather than episodes. As an addition consideration, it is common for too many positive alerts to be generated, especially when the underlying frequency of predictions is high. For example, consider a model which makes predictions every 5 minutes, warning a clinical team that a serious event may occur. Clinical evaluation and intervention may require far more than 5 minutes to complete. In this case, it makes sense to implement *snoozing*, or disabling of alerts for a certain period of time to allow for clinical workup. We propose that this *snoozing* be taken into account during the evaluation phase of the model as well, as this reflects the realized performance of an alert system, rather than a theoretical one. An example of this difference is illustrated in Figure 2. Formally, we can describe snoozing in the following way. Define s to be a nonnegative number which corresponds to the snoozing window. Let z be the threshold of choice and fix i to be the index corresponding to a patient who experiences an event. Then, assume t_{ij} is a valid positive prediction, *i.e.*, such that $h_\theta(X_{ij}) \geq z$. Any other index pair ik such that $t_{ik} \in (t_{ij}, t_{ij} + s]$ should be removed from any evaluation metric. Note that this should be done in a temporally consistent way, where the first positive prediction is chosen as the index prediction. Snoozing should be then silence any predictions which occur within the appropriate snoozing window, and the next valid positive prediction should begin the process again. This is illustrated in Figure 1B.

2.5.1 Consequences of Snoozing

By implementing snoozing as proposed, it is important to note that the precision is now no longer monotonic, as described in Fawcett (2006). As a consequence, we can no longer make the assumption that any instance that is classified positive with respect to a given threshold will be classified positive for all lower thresholds as well (Fawcett, 2006) since it may be snoozed at a different threshold. This leads to a significant increase in the computational complexity of computing metrics which utilize snoozing. We address this by implementing a native, cross-platform and open-source tool to compute metrics described in this work (`https://github.com/dihi/clinical-ts-metrics`). Our implementation outperforms a straightforward implementation using Python and Numpy by at least an order of magnitude in the worst case. Detailed benchmarks and implementation details can be found in the Appendix.

3 Recommended Metrics for Clinical Time Series

ROC analysis Metrics derived from the ROC are concerned with the trade-off between sensitivity and specificity. Since the clinical question of interest relating to sensitivity is about how often events are detected early, we recommend that these metrics be reported at the **episode** level.

Precision Recall (PR) analysis In precision-recall analyses, we are concerned with the trade-off between precision (positive predictive value) and recall. Here, we recommend that the precision be measured using the **prediction-level** true positive and false positive values (with snoozing if necessary from a clinical perspective) while the recall is measured at the **episode** level. This is a deviation from traditional calculations, as generally the recall is also computed at the prediction level. However, we argue that the clinically-relevant trade-off that PR curves can help examine should consider how many patients are captured at varying thresholds of the model, rather than considering prediction-level recall. A single point on the precision-recall curve now represents the explicit balance between how many true alerts and false alerts are generated a particular threshold versus how many events are captured sufficiently early.

4 Discussion

In this work, we discussed the early detection problem for clinical time series. Often, these problems are treated as binary classification problems and are therefore evaluated accordingly. However, using the ROC and Precision Recall curves often do not translate directly to the clinical utility of the models being discussed. Namely, the clinical utility of a model is tied to its ability to detect episodes of interest and the ratio of true alerts to false alerts that are introduced. Therefore, sensitivity and similar metrics should be evaluated at the episode level, while precision should be evaluated at the prediction level. Along with this, some models may generate far too many alerts for a clinical team to respond to. We also describe snoozing and provide an implementation for evaluating models under the context of snoozing. Our recommendations for how to present results for early detection models mirror current work in the literature which attempt to deal with these problems. One particular limitation of this work include how to deal with episodes where multiple events occur, which we hope will be addressed in future work.

References

Joseph Futoma, Sanjay Hariharan, and Katherine Heller. Learning to detect sepsis with a multitask Gaussian process RNN classifier. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1174–1182, Sydney, NSW, Australia, August 2017. JMLR.org.

Katharine E. Henry, David N. Hager, Peter J. Pronovost, and Suchi Saria. A targeted real-time early warning score (TREWScore) for septic shock. *Science Translational Medicine*, 7(299):299ra122, August 2015. ISSN 1946-6242. doi: 10.1126/scitranslmed.aab3719.

Soo Yeon Kim, Saehoon Kim, Joongbum Cho, Young Suh Kim, In Suk Sol, Youngchul Sung, Inhyeok Cho, Minseop Park, Haerin Jang, Yoon Hee Kim, Kyung Won Kim, and Myung Hyun Sohn. A deep learning model for real-time mortality prediction in critically ill children. *Critical Care*, 23(1):279, August 2019. ISSN 1364-8535. doi: 10.1186/s13054-019-2561-z. URL `https://doi.org/10.1186/s13054-019-2561-z`.

Nenad Tomašev, Xavier Glorot, Jack W. Rae, Michal Zielinski, Harry Askham, Andre Saraiva, Anne Mottram, Clemens Meyer, Suman Ravuri, Ivan Protsyuk, Alistair Connell, Cían O. Hughes, Alan Karthikesalingam, Julien Cornebise, Hugh Montgomery, Geraint Rees, Chris Laing, Clifton R. Baker, Kelly Peterson, Ruth Reeves, Demis Hassabis, Dominic King, Mustafa Suleyman, Trevor Back, Christopher Nielson, Joseph R. Ledsam, and Shakir Mohamed. A clinically applicable approach to continuous prediction of future acute kidney injury. *Nature*, 572(7767):116–119, August 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1390-1.

Stephanie L. Hyland, Martin Faltys, Matthias Hüser, Xinrui Lyu, Thomas Gumbsch, Cristóbal Esteban, Christian Bock, Max Horn, Michael Moor, Bastian Rieck, Marc Zimmermann, Dean Bodenham, Karsten Borgwardt, Gunnar Rätsch, and Tobias M. Merz. Early prediction of circulatory failure in the intensive care unit using machine learning. *Nature Medicine*, 26(3):364–373, March 2020. ISSN 1546-170X. doi: 10.1038/s41591-020-0789-4. URL `https://www.nature.com/articles/s41591-020-0789-4`. Number: 3 Publisher: Nature Publishing Group.

Alexander Lavin and Subutai Ahmad. Evaluating Real-time Anomaly Detection Algorithms - the Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44, December 2015. doi: 10.1109/ICMLA.2015.141. URL http://arxiv.org/abs/1510.03336. arXiv:1510.03336 [cs].

Jonathan A. Handler, Craig F. Feied, and Michael T. Gillam. Novel Techniques to Assess Predictive Systems and Reduce Their Alarm Burden. Technical Report arXiv:2102.05691, arXiv, July 2022. URL http://arxiv.org/abs/2102.05691. arXiv:2102.05691 [cs] type: article.

Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. Precision and Recall for Time Series. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/8f468c873a32bb0619eaeb2050ba45d1-Abstract.html.

Astha Garg, Wenyu Zhang, Jules Samaran, Savitha Ramasamy, and Chuan-Sheng Foo. An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2508–2517, June 2022. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2021.3105827. URL http://arxiv.org/abs/2109.11428. arXiv:2109.11428 [cs, stat].

A. G. Glaros and R. B. Kline. Understanding the accuracy of tests with cutting scores: the sensitivity, specificity, and predictive value model. *Journal of Clinical Psychology*, 44(6):1013–1023, November 1988. ISSN 0021-9762. doi: 10.1002/1097-4679(198811)44:6<1013::aid-jclp2270440627>3.0.co;2-z.

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.10.010. URL https://www.sciencedirect.com/science/article/pii/S016786550500303X.

A Appendix

A.1 Benchmark of Snoozed Metrics Implementation

The introduction of snoozing increases the complexity or computing the metrics described in this work. Practically speaking, implementations using traditional tools can be prohibitively slow, especially when evaluating over large test sets. In order to ameliorate this process, we implement the algorithm in Nim, which compiles code down to native binaries, which can be run on any major operating system (Mac OS, Windows, Linux, *etc*.). In order to benchmark the speed of the various implementations, we generated episode lengths uniformly from 1 time point to 3000 time points and evaluated the metrics at a varying number of thresholds. At each threshold and for each group of episodes, we calculate the confusion matrix summaries both at the encounter level and prediction level. The average number of seconds over three runs for each setting to complete these tasks is listed below.

Episodes	Thresholds	Python Implementation (s)	Our Implementation (s)
200	100	5.136	0.479
200	1000	51.177	2.890
200	10000	511.022	26.128
2000	100	55.655	4.963
2000	1000	558.562	30.362
2000	10000	5591.103	285.363

Our implementation is at least 10 times faster than the native python implementation and also includes time to read in the data and write out the results, so this is conservative estimate. Details on the implementation, source code, and benchmarking can be found at `https://github.com/dihi/clinical-ts-metrics`

Software Details All benchmarks were run on a Macbook Pro with M1 Pro chip. All versions of requisite libraries relevant to the benchmark are listed below

Software	Version
Python	3.8
Numpy	1.21.6
Nim	1.6.6
ArrayMancer	0.7.0