entropy

MDPI

Article

A Patient-Anchored, Oxygen Uptake Rate-Informed Gradient-Boosting Virtual Sensor for Arterial Oxygen Saturation Across a Nine-Hour Monitoring Gap

Renaldas Urniezius [1,*], Nicolas Augusto Pava Roldan [1], Vytautas Galvanauskas [1], Stefano Rinaldi [2], Tomas Kondratas [3], Lina Jankauskaite [3] and Vaidotas Gurskis [3]

[1] Department of Automation, Electrical and Electronics Faculty, Kaunas University of Technology, Studentu 48-320, LT-51367 Kaunas, Lithuania; vytautas.galvanauskas@ktu.lt (V.G.)
[2] Department of Information Engineering, University of Brescia, Via Branze 38, 25123 Brescia, BS, Italy; stefano.rinaldi@unibs.it
[3] Pediatric Department, Medical Academy, Lithuanian University of Health Sciences, A. Mickeviciaus g. 9, LT-44307 Kaunas, Lithuania; vaidotas.gurskis@lsmu.lt (V.G.)
* Correspondence: renaldas.urniezius@ktu.lt

Abstract

In intensive care, the arterial oxygen saturation of a ventilated patient is followed through the bedside monitor's pulse oximetry reading, which describes the present situation only; whether it will hold over the coming hours is left to the attention of the staff. Soft sensors exist for intermittently measured variables, but none can estimate the saturation hours ahead without a recent reading of it. We ask whether the continuously recorded ventilator and expired-gas signals can estimate saturation nine hours after the last available reading. Using 4017 hourly records of 31 ventilated children, we build a virtual sensor as three nested implementation scenarios of one code base: X0 (XGBoost 2.1.4 through our pipeline, no physics term active), XP (X0 plus a blend toward the last reading and the cumulative oxygen uptake rate over the gap per kilogram of body mass, derived from an oxygen balance of the arterial blood), and v4 (XP plus a patient-level anchor that recalibrates the nine-hour-old reading toward the patient's own history). Out-of-fold predictions under patient-grouped five-fold cross-validation are scored, with paired patient bootstrap intervals, on the 3730 h of 30 patients whose inputs contain no saturation value younger than nine hours. From X0 to XP, the mean per-patient MAPE decreases from 3.313% to 3.282%, resolved for this summary only; from XP to v4, it decreases to 3.115%, the pooled MAPE decreases from 3.885% to 3.729% (both resolved), the pooled correlation rises from 0.754 to 0.793, hypoxaemia below 90% at the end of the gap is detected with an AUC of 0.939 against 0.900 for the last reading, and the ensemble shrinks from 55 to 19 inputs and from 9637 to 474 leaves. An exact decomposition of every prediction shows that the two-coefficient anchor earns 82% of the improvement over persistence; most of the nine-hour information is the patient's own level, the gas-derived terms add a small but resolved contribution, and the sensor is a trend indicator rather than an alarm.

Keywords: virtual sensor; soft sensor; gradient boosting; XGBoost; physics-informed machine learning; mechanical ventilation; oxygen saturation; pulse oximetry; oxygen uptake rate; relative entropy; multivariate time series; explainability; patient-level calibration

Check for updates

Academic Editors: Cun Ji and Bo Li

Received: 24 September 2026
Revised: 16 October 2026
Accepted: 21 October 2026
Published: 24 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Although effective critical care depends on having timely and accurate knowledge of the patient's physiological state, intensive care units operate under persistent structural constraints that systematically undermine this requirement [1]. High patient-to-nurse ratios, equipment saturation during surge periods, and the physical impossibility of continuous bedside observation for every admitted patient mean that clinically significant changes in physiology can go undetected for extended periods [2]. Laboratory-based measurements remain the clinical standard for several critical physiological variables, including cardiac output [3] or arterial oxygen saturation, SaO_2, via co-oximetry [4,5]; yet, when the samples of intensive care patients are analysed in a central laboratory, the median turnaround time between blood collection and result availability is 56 min. This means that, in practice, some clinical decisions are grounded in delayed information [6]. At the bedside, the arterial saturation is followed continuously using pulse oximetry—the monitor's estimate of SaO_2 from the pulsatile absorbance of red and infrared light at a peripheral site, denoted SpO_2 in the clinical literature—and calibrated against co-oximetry [7,8]. The two quantities agree to within the oximeter's error of a few percentage points above 70%, with changes in one following changes in the other only moderately in the critically ill ($r = 0.606$ over 1085 paired readings [9]) and with occult hypoxaemia escaping the oximeter more often in patients with darker skin [10,11]. In this study, the monitor's value, registered hourly, is the quantity estimated; it is denoted S_aO_2m, the name of its column in our data, where the subscript m stands for the monitor.

Two kinds of absence must be distinguished in ICU time series because they call for different methods. The first is feature missingness: an input variable is sampled irregularly or not at all at some hours, and the pattern of its absence is itself informative, because tests are ordered when the staff expect them to be abnormal [12]; recurrent models with masking and decay [13], imputation strategies that remain available at deployment [14], and missing-indicator methods, with their caveats [15], address this kind of absence. The second is the *prolonged absence of the target variable itself*: the quantity to be estimated is not observed for hours, either because it is measured by an intermittent laboratory assay, the classical setting of the data-driven soft sensor in the process industry [16], or because the monitor's reading is to be anticipated before it is available. Estimating a blood gas between two draws [5] belongs to this second kind, and so does the task of this study: the saturation is to be estimated nine hours after the last available reading from signals that are never missing for long, so no imputation of the target is involved and the question is how much of the target the other signals carry.

This growing divergence between the current state and the most recent measurement is not an edge case but a structural property of any monitoring system that relies on intermittent invasive sampling [17,18]. Retrospective analyses of large ICU databases have documented that missing or delayed values in key physiological variables are systematically associated with suboptimal alarm management and delayed recognition of clinical deterioration [17,19]. For patients receiving mechanical ventilation, however, a continuously available high-resolution data source already exists: the ventilator itself, which continuously records a rich stream of respiratory signals that reflect the patient's respiratory mechanics and gas exchange capacity in real time [2,20]. The predictive value of this stream has already been established for within-ventilator prediction tasks. Qiu et al. [21] demonstrated that a long short-term memory relation network trained on MIMIC-III can predict patient-specific ventilator parameters in real time. If ventilator signals are sufficiently informative to predict future ventilator parameters, they should also carry information relevant to predicting other physiological variables, such as SaO_2, as demonstrated by prior work linking ventilator settings to patient oxygenation outcomes [22,23]. This fol-

lows the soft sensor paradigm, where continuously available signals are used to estimate intermittently measured variables [24].

In the medical domain, the soft sensor paradigm has been successfully applied across a broad range of continuous physiological estimation problems, establishing a strong precedent for the clinical feasibility of the approach [25]. Beyond oxygenation-related variables, soft sensor approaches have been successfully explored for glucose monitoring, cardiovascular assessment, respiratory monitoring, intracranial pressure estimation, and wearable health applications, consistently demonstrating the potential of machine learning models to infer clinically relevant physiological states from routinely collected signals [4,26–33].

Recent work in ICU time-series modelling emphasises the importance of explicitly representing temporal dynamics under irregular sampling. Lee et al. [18,34] showed that incorporating inter-event time gaps into recurrent architectures improves predictive performance by allowing models to exploit temporal spacing as an informative signal rather than treating it as missingness. Similarly, adaptive recurrent formulations that condition hidden state updates on elapsed time further improve the modelling of heterogeneous physiological trajectories in critical care settings [13]; temporal convolutional networks with spatial and temporal attention are the corresponding development in soft sensing [24].

Among physiological targets in critical care, oxygenation estimation has received growing attention; yet, existing approaches leave the core problem of this study unresolved. Woo et al. [35] trained Random Forest and XGBoost models on a large-scale multi-centre ICU dataset to predict SaO_2 from SpO_2 and clinical covariates, substantially reducing the estimation error over the pulse oximetry baseline, but the approach corrects an available, simultaneous SpO_2 signal rather than operating in its absence. Cabanas et al. [36] further highlight the regulatory and technical limitations of AI-enhanced pulse oximetry, reinforcing that SpO_2-dependent correction methods have fundamental constraints in failure scenarios.

The closest line of work to ours estimates oxygenation directly from ventilator signals without relying on pulse oximetry. Ghazal et al. [37] trained a bootstrap-aggregated decision tree to predict SpO_2 categories five minutes after a ventilator setting change, achieving an AUC below 0.75, a result that also reveals the core limitation: a five-minute classification horizon is far too narrow to address the multi-hour sensor-absence scenario that we target. Dong et al. [5] considered a closer approach, using a non-linear additive model to estimate the blood gas pH and $PaCO_2$ between invasive draws from ventilator-derived inputs, achieving clinically acceptable accuracy, but their target variable is not SaO_2 and the gap length is never varied or evaluated as a modelling parameter. Hser et al. [38] incorporated oxygenation estimation within a broader ICU respiratory monitoring system using gradient-boosted models, predicting respiratory failure events up to ten hours ahead, but their estimator still takes SpO_2 as a continuous input, making it inapplicable when pulse oximetry is unavailable. Finally, Si et al. [39] estimated SpO_2 from facial video using Random Forest and Support Vector Regression, demonstrating non-contact sensing potential but under controlled low-acuity conditions incompatible with the mechanically ventilated ICU patient.

Across these works, three gaps remain open: no approach reports prediction accuracy as a function of the elapsed time since the last available reading, none operates when no saturation value from the preceding hours is available, and none separates what a physical prior contributes from what the learned trees contribute under patient-level cross-validation. This study addresses these three gaps, building on the oxygen conservation law of our bioprocess soft sensors [40,41], on its transfer to human respiration [42,43], and on the maximum-relative-entropy weighting of our earlier estimators [41,44–46]. It estimates the monitored saturation nine hours after the last available reading from ventilator and gas analysis time series with gradient-boosted trees, using no saturation value younger than nine hours, and reports how the error grows from a four-hour to a nine-hour gap and how well a clinically relevant desaturation at the end of the gap is detected. The reason for not estimating the saturation from a recent reading alone is the purpose of the sensor: the blood gases of our cohort were rare, the gas analytics are recorded continuously, and a sensor that holds over nine hours from the gas analytics alone lets attention follow the tendency of a patient's saturation rather than its absolute level once the clinical procedures are settled, supports non-invasive supervision, and may in time reduce the number of blood samples drawn. The sensor is reported as three explicit implementation scenarios of one code base—the unmodified library through our pipeline (X0), the library with our anchor blend and the cumulative oxygen uptake rate over the gap (XP), and the latter with a patient-level calibrated anchor (v4), such that every improvement is attributed to the code that produced it and an exact decomposition of each prediction says how much of it is physics and how much is trees. The rest of the paper is organised as follows: Section 2 describes the data and the devices, the three scenarios, the entropy criterion of the loss, the physics-informed terms with their derivations, the boosted ensemble and the three places where our code enters it, and the evaluation protocol; Section 3 reports the three scenarios on identical folds with bootstrap intervals, the anchor coefficients, the anatomy of the predictions, the ablations, the detection of a low saturation, and the dependence on the age of the last reading and on the length of the gap; Sections 4 and 5 provide a discussion and conclusion, respectively; Appendix A presents the code listings and the equation-to-line mapping.

2. Materials and Methods

The objective of this study is to estimate S_aO_2m, the arterial oxygen saturation registered hourly from the bedside monitor's pulse oximeter, from hourly processed ventilator parameters and experimental gas analysis transducers, nine hours after the last available reading. A principal characteristic of the study is the translation of soft sensor technologies from bioprocess engineering to human physiological monitoring: as demonstrated by Kemesis et al. [43], industrial off-gas sensors can be adapted to the non-invasive continuous tracking of human respiratory gas exchange.

We define the dataset as the matrix $\mathbf{X} \in \mathbb{R}^{n_i \times n_m}$, where n_i is the number of hourly samples, n_m is the number of features that the boosted trees receive (its value depends on the scenario, Section 2.2), and the i-th row $\mathbf{x}_i \in \mathbb{R}^{n_m}$ is the feature vector of the i-th sample. The target vector is $\mathbf{y} \in \mathbb{R}^{n_i}$, the S_aO_2m values of every patient-hour. The mapping is

$$\hat{y}_i = \phi(\mathbf{x}_i, \theta) \tag{1}$$

where $\hat{y}_i$ is the predicted S_aO_2m of the i-th sample and $\phi(\cdot, \theta)$ is a regressor with learned parameters θ. Three implementations of ϕ are compared in this paper; they share one code base and differ in which of our additions are active.

2.1. Dataset and Preprocessing

The study cohort comprises $n_p = 31$ mechanically ventilated children treated in the paediatric intensive care unit of the Lithuanian University of Health Sciences Hospital Kauno Klinikos (Kaunas, Lithuania), totalling 4902 recorded hourly entries, Figure 1.

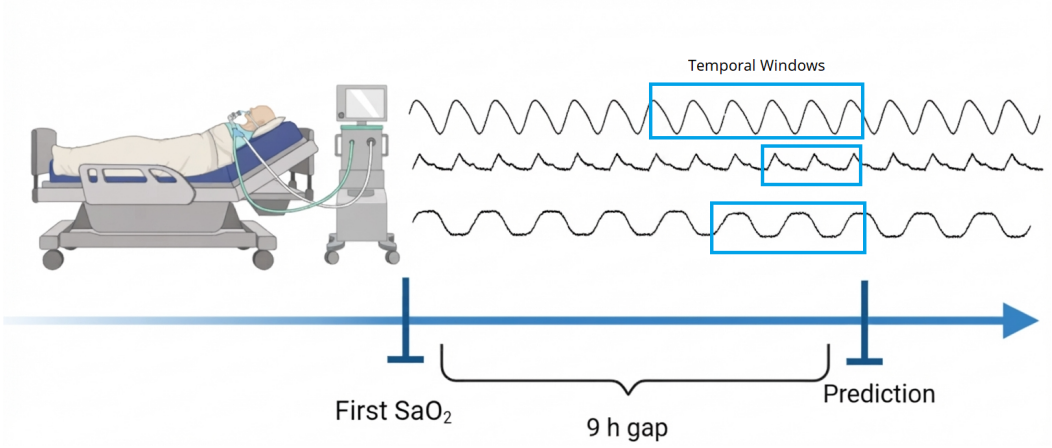


Figure 1. Temporal problem formulation and prediction horizon. The model estimates the current saturation y_t by bridging a nine-hour absence of the monitor's reading. The last available readings y_{t-9} and y_{t-10} supply the physiological baseline; the continuous ventilator and gas-sensor channels supply the real-time information available at hour t.

Of these, $n_i = 4017$ h have a registered target; all of them are predicted out-of-fold under five-fold patient-grouped cross-validation, and the results are reported on the 3730 h of 30 patients whose inputs contain no S_aO_2m value younger than nine hours (Section 3.7), with the values on all 4017 h listed alongside. The patient missing from the strict rows is not excluded from the study: patient 14 has 6 registered hours spanning 5 h, so none of its rows lies nine or more hours after its first reading; its rows are predicted out-of-fold and counted in the all-row figures. In 322 patient-hours (644 rows, 5 patients), two saturation values were documented within the same nominal hour. These rows are retained in the reported numbers; a sensitivity analysis keeping only the last value per hour is reported in Section 3.1.

2.1.1. Devices and Signals

The oxygen uptake rate signals were collected with the off-gas analysis device OUR120421A1 (JSC Cumulatis, Kaunas, Lithuania), the gas rig of Section 2.1.2, connected to the breathing circuit of the artificial lung ventilator (Servo-U, Maquet Critical Care, Solna, Sweden), whose settings and measured pressures and concentrations form the ventilator channels; the oxygen uptake rate signal is the one that our bioprocess soft sensors use [40,42,43]. The target S_aO_2m is the arterial oxygen saturation as observed with the pulse oximeter of the bedside patient monitor (Dash 4000, GE Healthcare) and written down by the nursing staff once per hour; in the clinical literature, this quantity is called SpO_2, and in our data, its column is named `Sa02m`: m for the monitor, a name that the derived columns keep (`Sa02m_lag_9h`, `Sa02m_first`, `runmean_Sa02m`). The saturation of the blood samples analysed on a Radiometer ABL90 analyser (column `Sa02` of our data) was registered only at the sampling times of the clinical routine, which were rare, and is neither an input nor the target of this study; it is the reference that the sensor is meant to need less often. No averaging, curation, or compensation was applied to the registered values before this study.

The target variable is discrete and integer-valued, spanning 72 distinct levels from 20% to 100% (median 96%, 1st percentile 52%; 350 observations fall below 85%). One value had been written down as 111% (patient 11; the neighbouring hours read 99, 100, and 98%), which lies outside the physical range of saturation and represents a documentation error in the manual record; in particular, the values were transcribed by the staff from the monitor without curation, and the oscillations and overshoot of the apparatus can bias a single reading. Before any lag, mean, or metric is formed, the pipeline now saturates every registered value to the physical range,

$$y_i \leftarrow \min\big(\max(y_i,\, 0),\, 100\big) \qquad (\%), \tag{2}$$

so that the row is kept with the value 100%; the rule is `target_valid_range` in `config.py` and is applied by `load_data` in `data_preparation.py` (Listing A1 in Appendix A), and every number in this paper is computed with Equation (2) in force. A detection of implausible volatility in the registered series is left to future work. This discrete distribution is directly relevant to the mechanics of the persistence baseline (Section 3.6). All features entering the decision trees were standardised using Z-score normalisation fitted strictly on the training folds; in contrast, all physiological quantities supplied directly to the analytical anchor (Section 2.4) were preserved in their raw, natural physical units.

We define a prediction horizon of nine hours: the longest prolonged stretch of a stay in which the attention of the physician is occasional rather than continuous, the night, so that a sensor which holds over nine hours can later serve a smart alarming system that calls attention when a patient's state requires it; Section 3.7 reports how the error grows from a four- to a nine-hour gap. The S_aO_2m history at $t-9$ and $t-10$ h is included to establish the patient's baseline prior to the simulated sensor gap, ensuring that the model remains grounded in the patient's physiological state without using future information. More precisely, the feature value provided at hour t represents the most recent observation recorded *at or before* $t-9$ h. Consequently, the actual elapsed time since that baseline reading was taken is at least nine hours and varies depending on clinical sampling irregularity; its empirical distribution is analysed in Section 3.6.

2.1.2. Signals and Their Lags in Each Scenario

Six real-time signals are engineered from the ventilator and the gas rig and lagged; three target-derived and two static descriptors complete the input. Table 1 lists the six signals with the lag hours that each scenario builds and, of those, the ones that the trees are allowed to see. Table 2 lists the remaining columns. The counts differ between the scenarios: the trees of X0 and XP receive $n_m = 55$ columns (six signals at hour t, their 44 lagged copies, and the five columns of Table 2 that are tree inputs), whereas the trees of v4 receive $n_m = 19$ (five signals at t with two lags each, plus four of those five columns). In v4, the pruning of the lag lists was decided by tolerance tests on the out-of-fold macro MAPE (Section 3); the signal `O2diffxPower` (a signal which is proportional to the oxygen uptake rate, OUR) is hidden from the trees in v4 because it feeds the physics terms instead.

Table 1. Dynamic signals and their lag hours. Each cell lists the hours t and $t-\ell$ at which the signal enters the data as a column (`data_preparation.py`); the count in brackets is the number of such columns. The names are those of the columns in the data file and in the code, where the oxygen is written with the digit after the letter O (`O2conc`, `O2diffxPower`, `O2xPress`, `O2xPressplusPEEP`, `CO2xPower`); they are typeset here with the subscript for readability. In X0 and XP, every column is a tree input. For $\mathrm{O_2}$`diffxPower`, the listed hours are also the sampling hours $\mathcal{L}$ of the cumulative oxygen uptake rate, Equation (17): the available values at these hours are averaged, and the average is multiplied by the gap D and divided by the body mass; in v4 these columns serve only that purpose and are not tree inputs. $\mathrm{O_2}$`conc`: inspired O_2 concentration at the mask inlet (%). `GM102B_Volts`: electrochemical gas-sensor output voltage (V). $\mathrm{CO_2}$`xPower`: expired CO_2 concentration times the ventilator power integral. $\mathrm{O_2}$`xPress`: $\mathrm{O_2}$`conc` times airway pressure. $\mathrm{O_2}$`xPressplusPEEP`: $\mathrm{O_2}$`conc` times pressure plus PEEP, a delivery composite. $\mathrm{O_2}$`diffxPower`: inlet-minus-expiratory O_2 concentration times the power integral, proportional to the oxygen uptake rate; it is the oxygen uptake rate signal $\mathrm{OUR}(t)$ of Section 2.4.

Signal	Hours in X0 and XP (All Tree Inputs)	Hours in v4	Of These, Tree Inputs in v4
$\mathrm{O_2}$`conc`	$t, t-1, 3, 4, 5, 6$ *(6 cols)*	$t, t-1, t{-}6$	$t, t-1, t{-}6$ *(3 cols)*
`GM102B_Volts`	$t, t-1, 2, 3, 4, 5, 7, 8$ *(8 cols)*	$t, t-1, t{-}8$	$t, t-1, t{-}8$ *(3 cols)*
$\mathrm{CO_2}$`xPower`	$t, t-1, 2, 3, 4, 5$ *(6 cols)*	$t, t-1, t{-}5$	$t, t-1, t{-}5$ *(3 cols)*
$\mathrm{O_2}$`xPress`	$t, t-1, 2, 3, 4, 5, 6, 8, 11, 12$ *(10 cols)*	$t, t-1, t{-}12$	$t, t-1, t{-}12$ *(3 cols)*
$\mathrm{O_2}$`xPressplusPEEP`	$t, t-1, 2, 4, 5, 6, 7, 8, 9, 10, 11$ *(11 cols)*	$t, t-1, t{-}11$	$t, t-1, t{-}11$ *(3 cols)*
$\mathrm{O_2}$`diffxPower`	$t, t-1, 2, 4, 5, 7, 8, 10, 11$ *(9 cols; also the sampling hours of Equation (17) in XP)*	$t, t-1, 2, 4, 7$ *(the sampling hours of Equation (17))*	none
tree inputs from this table: 50			15

Table 2. Target-derived and static columns. "Tree" describes whether the boosted trees receive the column in that scenario; "anchor" describes whether the column feeds the anchor of Section 2.4 instead. `runmean_SaO2m` exists only in v4.

Column	Description	X0, XP	v4
`SaO2m_lag_9h`	most recent S_aO_2m registered at or before $t-9$ h (%); the persistence value $y(\tau_i)$ of Equation (18)	tree and anchor	tree and anchor
`SaO2m_lag_10h`	most recent S_aO_2m registered at or before $t-10$ h (%)	tree	tree
`SaO2m_first`	the patient's first registered S_aO_2m (%)	tree	tree
`runmean_SaO2m`	mean of all S_aO_2m values registered at or before $t-9$ h, Equation (21)	not built	anchor only
`count_hours`	elapsed treatment time (h)	tree	tree
`Svoris`	body mass (kg); also normalises the cumulative OUR term	tree	anchor scale only
tree inputs from this table		5	4
total tree inputs n_m		55	19

2.2. Three Implementation Scenarios

Every number in this paper is attributed to one of three scenarios of one code base (Python (v4) pipeline and a patched XGBoost 2.1.4 C++ library). They are nested: each adds to the previous one and changes nothing else.

- **X0**—The XGBoost 2.1.4 library, trained by our Python pipeline with the tree inputs of Tables 1 and 2 ($n_m = 55$), the initial prediction equal to the persistence value $y(\tau_i)$, the cost-sensitive weights of Section 2.3. The two C++ terms added for XP are also compiled into the library used here, but both coefficients are zero ($\beta = \beta_1 = 0$ in Equation (34)), which makes them identities: X0 produces what unmodified XGBoost 2.1.4 produces from the same pipeline, up to float32 rounding.
- **XP—XGBoost with our anchor blend and cumulative OUR term.** X0 plus the two C++ terms of Section 2.6 with their configured coefficients $\beta = 0.1418$ and $\beta_1 = 0.00139$: the final prediction is pulled toward the persistence value and receives the cumulative oxygen uptake rate over the gap per kilogram of body mass, Equation (17). Same trees, same $n_m = 55$.
- **v4—XP with the patient-level calibrated anchor.** XP plus the anchor of Section 2.4.5, which replaces the raw persistence value with a least-squares combination of the persistence value and the patient's own measurement history, fitted on each training fold; its trees receive $n_m = 19$ inputs and the hyperparameters. $\beta = 0.3677$, $\beta_1 = 0.00124$.

Every code excerpt and equation in this paper follows a unified typographical convention. In the code listings, dark grey denotes unmodified upstream XGBoost (v2.1.4) source code; regular black font indicates our additions active in both XP and v4 (our Python pipeline and C++ extensions); and **bold font** denotes additions active exclusively in v4. Two listings stay in the text, the patient-level anchor of v4 (Listing 1) and the final prediction in the C++ library (Listing 2); the others, with the table that follows every term of the final prediction through the code and the equation-to-line mapping, are collected in Appendix A. The lines of original XGBoost source that appear inside our listings are rendered in grey, and no listing in this paper changes a line of the original algorithm: our code enters before the first tree is grown (the anchor and the weights) and after the last one (the final prediction), as Section 2.5 states.

In the equations, our contributions carry the label [XP] if active in both XP and v4, and are set in bold with the label [v4] if unique to v4. Standard equations from baseline XGBoost or established physiological literature remain unmarked. Table 3 lists every mathematical symbol with its unit and corresponding code variable.

2.3. The Loss: Cost-Sensitive Sample Weights and the Relative-Entropy Criterion

Two fits in this paper minimise a weighted sum of squared errors over the training rows: the least-squares fit of the patient-level anchor of v4 (Section 2.4.5) and the boosted trees of all three scenarios (Section 2.5). Both use one dimensionless weight per row,

$$w_i = \frac{1 - k_w}{|y_i|^{\nu} + \varepsilon} + k_w, \tag{3}$$

where y_i is the target in %, $k_w \in [0, 1]$ is a dimensionless floor (`custom_kp`), $\nu \geq 0$ is a dimensionless exponent (`custom_power`), and $\varepsilon = 10^{-6}$ is a stabiliser in $\%^{\nu}$ (`custom_eps`); the three are fields of `config.py`, class `TrainConfig`, and the weights are the array `sw` computed by `compute_sample_weights` (`utils_prediction.py`). The weights are the same in the three scenarios up to the values of k_w and ν.

The family (3) descends from our maximum-relative-entropy estimators [40,41,44,45], where it was written for the biomass or product concentration of a cultivation with a fixed

square in place of the exponent ν, and its single coefficient was named K_{exp}; in the notation of this paper, that coefficient is k_w and the observed quantity is y_i. The principle behind the family is that of maximum relative entropy: a prior density is updated to the posterior that is closest to it, in the sense of the Kullback–Leibler divergence [47], among the densities consistent with the constraints at hand [48]; the Kalman filter and the joint estimation of states and parameters under differential equation constraints follow from it [49,50]. Applied to a fit, the principle makes the relative entropy between the density of the prediction and the likelihood of the observation the objective of the estimation. In [44], the variance of the Gaussian likelihood was set equal to the square of the model's local mean, which turns that objective into a squared relative error, the $k_w \to 0$ end of the family. In [45] the posterior of a sample was written as a Gaussian of constant variance, the likelihood as a Gaussian centred on the observed value with an individual variance, their relative entropy as the likelihood function of the fit, and the constant of that expression was replaced by a tuning coefficient K_{exp} which trades the least-squares criterion against the mean-absolute-percentage-error criterion: the variance of the observation, the nuisance parameter of the fit, is thereby made inversely proportional to the weight of its squared error, as shown in Equation (5) below. The exponent ν replaced the square in [46], on the volatility of financial series, to make the family generic, and is used here with the stabiliser ε. For a Gaussian density P_{post} of the prediction $\hat{y}_i$ and a Gaussian likelihood P_{lik} centred on the observation y_i with variance σ_i^2 (in $\%^2$), the relative entropy of row i is [41]

$$S_i = -\int P_{post}(\hat{y}) \, \ln \frac{P_{post}(\hat{y})}{P_{lik}(\hat{y})} \, \mathrm{d}\hat{y} = -\frac{(\hat{y}_i - y_i)^2}{2\,\sigma_i^2} + \text{const}, \tag{4}$$

which is dimensionless, with the variance model [41]

$$\sigma_i^2 = \sigma_0^2 \, \frac{|y_i|^\nu}{1 - k_w + k_w \, |y_i|^\nu}, \tag{5}$$

in which σ_0^2 is a constant scale in $\%^2$, y_i enters with its numerical value in %, and $\nu = 2$ gives the printed form of the cited papers. Inserting Equation (5) into Equation (4) and summing over the training rows gives

$$-2 \sum_i S_i = \frac{1}{\sigma_0^2} \sum_i \left[\frac{1 - k_w}{|y_i|^\nu} + k_w \right] (y_i - \hat{y}_i)^2 + \text{const}, \tag{6}$$

whose bracket is Equation (3) with $\varepsilon = 0$. The weighted squared loss used below is therefore the relative-entropy criterion of [41,45], generalised by the exponent ν and the stabiliser ε [46]; k_w is the single coefficient that trades the error relative to $|y_i|^\nu$ (the MAPE end, $k_w \to 0$) against the squared error (the variance end, $k_w \to 1$), and Equation (5) is the statement that the variance assigned to an observation is inversely proportional to its weight, $\sigma_i^2 = \sigma_0^2 / w_i$ for $\varepsilon = 0$. The two coefficients k_w and ν are the only places where the loss of the trees is shaped; they were tuned together with the tree hyperparameters (Section 2.8.7), and their measured effect on this cohort is given in Section 3.5.

Two remarks clarify what k_w does in the boosted trees, where the weights enter the gradients of Equation (29) rather than a closed-form regression. The relative weights decide how much a low-saturation row counts against a high-saturation one; when $(1 - k_w)/|y_i|^\nu$ is small against k_w for every row, as it is for the tuned values of this paper (Section 3.5), the weights are constant to that precision and the criterion is the squared error. The common scale k_w is nevertheless not inert for the trees: a constant weight multiplies every gradient and Hessian sum of Equations (31)–(33) by k_w, so the fitted trees are exactly those of the unweighted model with the penalties λ / k_w and α / k_w and a mini-

mum child weight of `min_child_weight`/k_w rows (the early-stopping RMSE is scale-free, so the number of trees does not change), whereas the anchor regression of Equation (22), invariant to a common factor, is unaffected. The tuned floors are therefore part of the trees' regularisation; their effective values are given in Section 3.5.

Table 3. Symbols units, and names in the code (`dp`: `data_preparation.py`; `up`: `utils_prediction.py`; `lc`: `src/learner.cc`). a.u.: arbitrary units of the gas signals.

Symbol	Meaning	Unit	Name in the Code
i, n_i	row (one patient-hour), number of rows	—	row of the data frame
$p(i)$, n_p	patient of row i, number of patients	—	`patient_id`
t_i	hour of row i	h	`LocalTime`
y_i	registered S_aO_2m (the monitor's pulse oximetry value), the target	%	`SaO2m`
D	nominal gap	h	`virtual_sensor_gap` $= 9$
τ_i, D_i	time and age of the last measurement available at t_i, Equation (18)	h	inside `_time_based_lag_last_known` (dp)
$y(\tau_i)$	last available measurement	%	`SaO2m_lag_9h`; `raw_lag_tr`, `raw_lag_va` (up)
$\mathcal{H}_i$; $\bar{y}_{\mathcal{H}_i}$	measurement times available at $t_i - D$; mean of y over them, Equation (21)	h; %	`runmean_SaO2m`
m_i^{kg}	body mass	kg	`Svoris`
OUR(t)	oxygen uptake rate signal at the airway	a.u.	`O2diffxPower`
CPR(t)	carbon dioxide production rate signal	a.u.	`CO2xPower`
k_{OUR}, k_{CPR}	coefficients of the balance, Equation (14)	% kg h^{-1} a.u.$^{-1}$	—
$\mathcal{L}$	lag set of the OUR signal	h	lags of `O2diffxPower`
cOUR$_i$	cumulative oxygen uptake rate over the gap per kg, Equation (17)	a.u. h kg^{-1}	`integral_O2diffxPower`; field `custom_integral`
$\hat{y}_{0,i}$	initial prediction, the anchor	%	`raw_base_train`, `raw_base_val`; `base_margin`; `anchor` (lc)
c_0; c_1, c_2	coefficients of the patient-level anchor, Equation (21)	%; —	`coef[0]`; `coef[1]`, `coef[2]` (up)
w_i	sample weight, Equation (3)	—	`sw`
k_w; ν; ε	floor, exponent, and stabiliser of w_i, Equation (3)	—; —; %$^{\nu}$	`custom_kp`, `custom_power`, `custom_eps`
b, n_b	boosting round, number of trees	—	rounds of `xgb.train`
f_b, $s_{b,j}$	tree b, score of its leaf j	%	XGBoost internal
η	learning rate	—	`learning_rate`
λ; α; γ	L2, L1, and per-split penalties	—; %; %^{2}	`reg_lambda`, `reg_alpha`, `gamma`
$g_{b,i}$, $h_{b,i}$	gradient and Hessian of row i in round b	%, —	XGBoost internal
$\hat{y}_{b,i}$; $\hat{y}_{n_b,i}$	running and raw boosted prediction	%	prediction buffer; `preds` (lc)
$\tilde{y}_i$	final prediction, Equation (34)	%	`pred_val`, `pred_tr` (up)
β; β_1	anchor blend; weight of cOUR$_i$	—; % kg h^{-1} a.u.$^{-1}$	`custom_b`, `custom_b1`
K_La; $pO_2{}^*$, $PaO_2{}^*$	precursor model: mass transfer capacity; maximal oxygen solubility	s^{-1}; mmHg	—

2.4. From the Oxygen Balance to the Physics-Informed Terms

This subsection derives, from the oxygen conservation law of our earlier work, the two terms that the scenarios add to the boosted trees of Section 2.5: the cumulative oxygen uptake rate of XP, Equation (17), and the patient-level anchor of v4, Equation (21). The derived columns `integral_O2diffxPower` and `runmean_SaO2m` are removed from the tree inputs (`utils_prediction.py`, `_extract_base_margin`) and reach the model only through these terms.

The idea that organises the subsection is that of an *integration part*. A signal need not enter the model as one more input column of the trees if it is a quantity that the balance lets us integrate in closed form over the gap; here, the change in the saturation accumulated from the oxygen uptake and the recalibration of the last reading toward the patient's own level are computed outside the trees and enter the prediction as their own share, so that the trees are left with what the balance does not explain. Each such term shrinks the black box and raises the explainability of this semi-supervised learning: the trees of v4 receive 19 inputs instead of 55 and end with 474 leaves instead of 9637 (Section 3.1), and an exact decomposition says how much of every prediction came from the closed-form terms and how much from the trees (Section 2.7); in later work, the same mechanism is the natural place for the reinforcement of the trees by a physical term. The device of Section 2.1 measures the oxygen uptake rate; the balance also contains a consumption term driven by the carbon dioxide production, which is derived below because it closes the balance and is not active in any scenario of this paper ($k_{CPR} = 0$ in Equation (14)). This is why the title names the oxygen uptake rate rather than the balance as a whole.

2.4.1. Precursors: The Oxygen Conservation Law of Our Earlier Work

In aerobic bioprocesses, in a stationary oxidative state, the oxygen uptake rate of the culture, OUR, equals the oxygen transfer rate from the gas, OTR, in both the amount of oxygen per unit volume and time [40,42]:

$$\mathrm{OUR} = \mathrm{OTR}. \tag{7}$$

In the time domain, the same conservation law reads [42,51]: the dissolved oxygen, written as its partial pressure $p\mathrm{O}_2$ (mmHg), accumulates by what is transferred minus what is taken up,

$$\frac{\mathrm{d}p\mathrm{O}_2(t)}{\mathrm{d}t} + \mathrm{OUR}(t) = \mathrm{OTR}\big(p\mathrm{O}_2(t), t\big), \tag{8}$$

and the transfer is proportional to the distance from the maximal oxygen solubility $p\mathrm{O}_2{}^*$ (mmHg), with the oxygen mass transfer capacity $K_L a$ (s^{-1}) [42]:

$$\mathrm{OTR}\big(p\mathrm{O}_2(t), t\big) = K_L a \left(p\mathrm{O}_2{}^*(t) - p\mathrm{O}_2(t)\right). \tag{9}$$

Substituting Equation (9) into Equation (8) and replacing the dissolved oxygen in the nutritional medium with the partial pressure of oxygen in arterial blood, PaO_2 (mmHg), Survyla et al. [42] and Kemesis et al. [43] obtained the model of arterial oxygen, which they applied to a human volunteer and to ventilated children,

$$\frac{\mathrm{dPaO}_2(t)}{\mathrm{d}t} = K_L a \left(\mathrm{PaO}_2{}^*(t) - \mathrm{PaO}_2(t)\right) - k_{OUR}\,\mathrm{OUR}(t), \tag{10}$$

in which $\mathrm{OUR}(t)$ is the airway signal of their gas analysis device and k_{OUR} its calibration coefficient, so that $k_{OUR}\,\mathrm{OUR}(t)$ is in $\mathrm{mmHg\,s^{-1}}$ (in [42]: $k_{OUR} = 1.408$, $K_L a = 0.4217$ s^{-1}, $\mathrm{PaO}_2{}^* = 120.2$ mmHg for one volunteer). To use sampled data, Survyla et al. [42] integrated the balance between two neighbouring sample times t_{i-1} and t_i (s) and divided

both sides by the interval $\Delta t_{i,i-1} = t_i - t_{i-1}$, a bar denoting the average over the interval (their Equations (8) and (9)); applied to Equation (10), this step gives

$$\frac{\mathrm{PaO_2}(t_i) - \mathrm{PaO_2}(t_{i-1})}{\Delta t_{i,i-1}} + k_{OUR}\,\overline{\mathrm{OUR}} = K_L a\,\left(\overline{\mathrm{PaO_2}^*} - \overline{\mathrm{PaO_2}}\right). \tag{11}$$

Integrating the rate before using it is also the idea of the cumulative oxygen uptake rate cOUR [40]: the integral of OUR is less noisy than OUR itself. Two features of Equation (10) do not fit the present task. It is written for partial pressure and, in the trial of [42], the haemoglobin saturation did not change noticeably, so the dissociation curve played no role; in contrast, here, the target is the saturation itself. It needs $K_L a$ and $\mathrm{PaO_2}^*$, which were identified per person from blood gases, whereas no per-patient calibration is available at prediction time here.

2.4.2. The Balance Used Here: Saturation, Whole-Body Amounts, Body Mass on the Left

Three physiological facts connect Equation (8) to the saturation y (%). The oxygen content of arterial blood, $\mathrm{C_aO_2}$ (mL $\mathrm{O_2}$ per dL of blood), is carried almost entirely by haemoglobin, of concentration Hb (g/dL) [52]:

$$\mathrm{C_aO_2} = 1.34\,\mathrm{Hb}\,\frac{y}{100} + 0.003\,\mathrm{PaO_2}; \tag{12}$$

at Hb = 14 g/dL, y = 95%, and $\mathrm{PaO_2}$ = 100 mmHg, the dissolved term is 0.3 against 17.8 mL/dL bound to haemoglobin. The amount of oxygen in the circulating blood is the content times the blood volume V_b (dL), and the blood volume is proportional to the body mass m^{kg} (kg), $V_b = k_b\,m^{kg}$, with $k_b \approx 0.7$ dL/kg in adults [53]. Written for amounts instead of concentrations, Equation (8) states that the oxygen held in the blood changes with the oxygen taken from the ventilator gas, $\dot{V}_{\mathrm{O_2}}^{air}$ (mL $\mathrm{O_2\,h^{-1}}$), which is what the airway signal measures (inspired minus expired oxygen times the gas flow [54]), minus the oxygen consumed by the tissues, which equals the carbon dioxide production $\dot{V}_{\mathrm{CO_2}}$ (mL $\mathrm{CO_2\,h^{-1}}$) divided by the dimensionless respiratory quotient RQ. With body mass on the left:

$$k_b\,m^{kg}\,\frac{\mathrm{dC_aO_2}(t)}{\mathrm{d}t} = \dot{V}_{\mathrm{O_2}}^{air}(t) - \frac{\dot{V}_{\mathrm{CO_2}}(t)}{\mathrm{RQ}}. \tag{13}$$

Equation (13) is the conservation law of Equation (8), as is Equation (10) and its sampled form Equation (11); Table 4 lists how each term of Equation (11) corresponds to a term of Equation (13) and of its integrated form, Equation (16) below. The difference is the closure. Equation (11) models the transfer from the gradient, $K_L a(\mathrm{PaO_2}^* - \mathrm{PaO_2})$, and uses the airway signal as the uptake, the two being equal in the stationary state of Equation (7). Equation (13) takes the transfer into the blood from the airway signal, which is what that signal physically measures, and takes the tissue consumption from the $\mathrm{CO_2}$ production; it therefore needs neither $K_L a$ nor $\mathrm{PaO_2}^*$.

The assumptions are as follows: (A1) the arterial blood is one well-mixed compartment whose volume $k_b\,m^{kg}$ does not change over the gap; (A2) Hb is constant over the gap, and the dissolved term of Equation (12) is neglected, so $\mathrm{dC_aO_2}/\mathrm{d}t = (1.34\,\mathrm{Hb}/100)\,\mathrm{d}y/\mathrm{d}t$; (A3) the airway signal $\mathrm{OUR}(t)$, the column `O2diffxPower` (inlet-minus-expiratory $\mathrm{O_2}$ concentration times the ventilator power integral, in arbitrary units a.u.), is proportional to the oxygen taken from the gas, $\dot{V}_{\mathrm{O_2}}^{air} = c_{OUR}\,\mathrm{OUR}(t)$ with c_{OUR} in mL $\mathrm{O_2\,h^{-1}}$ a.u.$^{-1}$, the power integral standing for the gas flow; (A4) the signal $\mathrm{CPR}(t)$, the column `CO2xPower` (a.u.), is proportional to the $\mathrm{CO_2}$ production, $\dot{V}_{\mathrm{CO_2}} = c_{CPR}\,\mathrm{CPR}(t)$ with c_{CPR} in mL $\mathrm{CO_2\,h^{-1}}$ a.u.$^{-1}$, and RQ is constant over the gap; (A5) these constants are shared by the patients of a training fold; (A6) the hourly samples are missing at random; (A7) the monitor's pulse oxime-

try reading, the target y (Section 2.1), follows the arterial saturation of the blood to within the error of the oximeter, a few percentage points above 70% and more below it [7,55], with changes in the two agreeing only moderately in the critically ill [9]; the balance, written for the arterial blood, is therefore applied to the registered reading, and the error of (A7) is part of the error of the sensor. Substituting (A2)–(A4) into Equation (13) and collecting the constants into $k_{OUR} = 100\, c_{OUR}/(1.34\,\mathrm{Hb}\,k_b)$ and $k_{CPR} = 100\, c_{CPR}/(1.34\,\mathrm{Hb}\,k_b\,\mathrm{RQ})$, both in $\%\,\mathrm{kg\,h^{-1}\,a.u.^{-1}}$, gives the balance with body mass on the left,

$$m^{kg}\,\frac{\mathrm{d}y(t)}{\mathrm{d}t} = k_{OUR}\,\mathrm{OUR}(t) - k_{CPR}\,\mathrm{CPR}(t), \tag{14}$$

and dividing both sides by m^{kg},

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = k_{OUR}\,\frac{\mathrm{OUR}(t)}{m^{kg}} - k_{CPR}\,\frac{\mathrm{CPR}(t)}{m^{kg}}. \tag{15}$$

This is why the oxygen uptake signal enters the code per kilogram of body mass. In the three scenarios of this study, the consumption term of Equation (15) is not used: its coefficient k_{CPR} is 0 there, and only the uptake term is carried from here on. The consumption term is nevertheless kept in the derivation because it is the closure of the balance: the airway signal measures what enters the blood, the carbon dioxide production measures what the tissues take, and a sensor that integrates both would predict the change in the saturation from their difference. With $k_{CPR} = 0$, the scenarios integrate the uptake alone, and whatever the consumption contributes is left to the trees, which receive the `CO2xPower` columns of Table 1; fitting k_{CPR} is the next closed-form term to move out of the black box, and no ablation of it is possible here because no scenario in this paper fitted it.

Table 4. Correspondence between the precursor balance, Equation (11), and the balance of this paper, Equations (13) and (16).

Role in the Balance	**Precursor, Equation (11)**	**This Paper, Equations (13) and (16)**
state variable	PaO_2 (mmHg), per unit volume	y (%) through C_aO_2, Equation (12), whole body through $k_b\,m^{kg}$
accumulation	change in PaO_2 over $\Delta t_{i,i-1}$	change in y over the gap D
transfer into the blood	$K_L a(\overline{PaO_2{}^*} - \overline{PaO_2})$, modelled	$\dot{V}_{O_2}^{air}$, measured at the airway as $\mathrm{OUR}(t)$
consumption	$k_{OUR}\,\overline{\mathrm{OUR}}$, from the airway signal	$\dot{V}_{CO_2}/\mathrm{RQ}$, from $\mathrm{CPR}(t)$; not used in the three scenarios ($k_{CPR} = 0$)
time step	interval between two samples	gap $D = 9$ h

2.4.3. Integration over the Gap: The Cumulative Oxygen Uptake Rate

We integrate Equation (15) over the gap $(t_i - D,\ t_i]$, as Equation (11) integrates between samples, with t_i the hour of row i and $D = 9$ h (`config.py`, `virtual_sensor_gap`):

$$y(t_i) - y(t_i - D) = k_{OUR} \int_{t_i - D}^{t_i} \frac{\mathrm{OUR}(s)}{m_i^{kg}}\mathrm{d}s, \tag{16}$$

where m_i^{kg} is the last known body mass of the patient in row i (column `Svoris`; for a patient with no recorded mass, the mean over the patients). The integral is the cumulative oxygen uptake rate of [40], taken over the gap and per kilogram. The code evaluates it with the rectangle rule from the OUR values recorded at the hours of the lag set $\mathcal{L}$ (h), the lags of Table 1 plus the current hour ($\mathcal{L} = \{0, 1, 2, 4, 5, 7, 8, 10, 11\}$ in X0 and XP, $\{0, 1, 2, 4, 7\}$ in v4): the available values are summed, the sum is divided by their number $|\mathcal{L}_i|$ ($\mathcal{L}_i \subseteq \mathcal{L}$, the hours at which a value exists for row (i), and the average is multiplied by the width D of the integration interval and divided by the body mass:

$$\mathrm{cOUR}_i = \frac{D}{m_i^{kg}} \cdot \frac{1}{|\mathcal{L}_i|} \sum_{\ell \in \mathcal{L}_i} \mathrm{OUR}(t_i - \ell) \;\approx\; \int_{t_i - D}^{t_i} \frac{\mathrm{OUR}(s)}{m_i^{kg}}\, \mathrm{d}s \qquad [\mathrm{XP}], \tag{17}$$

in a.u. h kg^{-1} (column `integral_O2diffxPower`, Listings A2 and A3 in Appendix A; the mean over the available columns is `mean(axis=1, skipna=True)`). Under (A6), the sample mean is unbiased for the time average of OUR$/m^{kg}$ over the gap, so Equation (17) is unbiased for the integral [56]; the plain sum $\sum_\ell \mathrm{OUR}(t_i - \ell) \cdot 1\,\mathrm{h}$ would equal $|\mathcal{L}_i|/D$ times it and shrink with every missing hour. If OUR$/m^{kg}$ is Lipschitz with constant Λ (a.u. kg^{-1} h^{-1}), the error of the rectangle rule on an hourly grid is at most $\frac{1}{2}\,\Lambda\, D \cdot 1\,\mathrm{h}$ [57]. The factor is the gap D and not the number of columns; the two coincide in X0 and XP when all nine values are present ($|\mathcal{L}_i| = D = 9$) but not in v4, and the factor D keeps the units when lags are pruned. Only the row's own signals up to t_i are used.

2.4.4. The Last Available Reading

The reading of y that the sensor may use at hour t_i is the most recent one registered at or before $t_i - D$:

$$\tau_i = \max\{\, s \le t_i - D : y(s) \text{ observed for patient } p(i) \,\}, \qquad D_i = t_i - \tau_i \ge D, \tag{18}$$

where D_i denotes the actual elapsed time (in hours). The term $y(\tau_i)$ corresponds to the nine-hour lagged oxygen saturation (`SaO2m_lag_9h`), representing the most recent measurement recorded at or before $t_i - D$. During the initial hours of an ICU admission before a nine-hour history is established ($t_i < D$), this value defaults to the patient's first recorded measurement. Substituting $y(\tau_i)$ for $y(t_i - D)$ and substituting Equation (17) into Equation (16) yields the integrated oxygen mass balance,

$$y_i \approx y(\tau_i) + k_{OUR}\, \mathrm{cOUR}_i, \tag{19}$$

whose two terms the scenarios use as listed in Table 5. In X0 and XP, the initial prediction of the boosted trees is the last available measurement itself,

$$\hat{y}_{0,i} = y(\tau_i) \qquad (\mathrm{X0},\ \mathrm{XP}), \tag{20}$$

so the trees learn corrections to the nine-hour-old reading rather than to a constant.

β_1 is found by tuning, not by least squares, and in X0 and XP, the trees also receive the OUR signal columns, so β_1 is not an estimate of k_{OUR}; both multiply the cumulative oxygen uptake per kilogram over the gap.

Table 5. How each scenario uses the terms of the integrated oxygen balance, Equation (19).

Term	X0	XP	v4
$y(\tau_i)$	initial prediction, Equation (20)	initial prediction, Equation (20)	recalibrated toward the patient's history, Equation (21)
$k_{OUR}\,\mathrm{cOUR}_i$	not used	$\beta_1\,\mathrm{cOUR}_i$ in Equation (34), $\beta_1 = 0.00139$	$\beta_1\,\mathrm{cOUR}_i$ in Equation (34), $\beta_1 = 0.00124$

2.4.5. Scenario v4: The Patient-Level Anchor

Let $\mathcal{H}_i = \{\, s \le t_i - D : y(s) \text{ observed for } p(i) \,\}$ be the times of all measurements of the patient available at $t_i - D$; this is the same information as $y(\tau_i)$ uses, so no new leakage enters. In v4, the initial prediction recalibrates the last available measurement toward the patient's own past mean:

$$\hat{y}_{0,i} = c_0 + c_1\, y(\tau_i) + c_2\, \bar{y}_{\mathcal{H}_i}, \qquad \bar{y}_{\mathcal{H}_i} = \frac{1}{|\mathcal{H}_i|} \sum_{s \in \mathcal{H}_i} y(s) \qquad [\text{v4}] \tag{21}$$

with c_0 in % and c_1, c_2 dimensionless, and $\bar{y}_{\mathcal{H}_i} := y(\tau_i)$ when $\mathcal{H}_i$ is empty (the first D hours of a stay; `utils_prediction.py`). $\bar{y}_{\mathcal{H}_i}$ is the column `runmean_Sa02m` (`data_preparation.py`, `compute_target_runmean`). The three coefficients are the weighted least-squares solution on the training rows $\mathcal{T}$ of the fold, with the weights w_i of Equation (3):

$$(c_0, c_1, c_2) = \arg\min \sum_{i \in \mathcal{T}} w_i \left(y_i - \hat{y}_{0,i}\right)^2 \qquad [\text{v4}] \tag{22}$$

with $\hat{y}_{0,i}$ defined by Equation (21). In the implementation, the three anchor coefficients (c_0, c_1, c_2, representing the intercept, persistence weight, and historical patient mean) are solved in Python and transferred via the `base_margin` field of the training and validation `DMatrix` objects. In the C++ core, this directly populates `base_margin_` ($\hat{y}_{0,i}$), which acts as the initial reference score for each row in the tree recursion (Section 2.5) and the physiological anchor of the final ensemble prediction in Equation (34).

Each cross-validation fold estimates these coefficients strictly using its training cohort and applies them out-of-fold to the held-out validation patients. The resulting fold-averaged coefficients are $c_0 = 3.379$, $c_1 = 0.2480$, and $c_2 = 0.7161$ (Section 3). Because $c_1 + c_2 = 0.964$, Equation (21) represents, to within four per cent, a weighted combination between the last available reading (26%) and the patient's long-term historical baseline (74%). The theoretical rationale for assigning a weight strictly below unity to $y(\tau_i)$ is regression attenuation [58,59]: framing the observed measurement as $y(\tau_i) = \mu_i + \xi_i$, where μ_i is the true underlying physiological state and ξ_i represents independent measurement noise, the least-squares shrinkage slope on $y(\tau_i)$ is

$$c_1^* = \frac{\mathrm{Var}(\mu)}{\mathrm{Var}(\mu) + \mathrm{Var}(\xi)} < 1, \tag{23}$$

and $\bar{y}_{\mathcal{H}_i}$ carries the same level with less noise, so Equation (22) moves weight from $y(\tau_i)$ to $\bar{y}_{\mathcal{H}_i}$, a shrinkage toward the patient's own level [60]. The coefficients of every fold, with the standard error of their sum and their out-of-fold accuracy, are examined in Section 3.2, including the one fold in which $c_1 + c_2$ exceeds one. The running mean itself is computed by `compute_target_runmean` (Listing A4 in Appendix A); the fit and its handover to the library are in Listing 1.

Listing 1: Scenario v4: the patient-level anchor, Equations (21) and (22), in `utils_prediction.py`. In `build_calib_design`, `m_tr` and `m_va` hold $y(\tau_i)$ of the training and validation rows, the column of ones is the one c_0 multiplies, and `"runmean_SaO2m"` is $\bar{y}_{\mathcal{H}_i}$. In `_predict_fold`, `sw` holds w_i, line 321 is Equation (22) (`coef` $= (c_0, c_1, c_2)$), lines 323–324 are Equation (21) for the training and the validation rows, and lines 361–371 hand the result to XGBoost as the `base_margin` of both matrices (`raw_base_train`, `raw_base_val` hold $y(\tau_i)$ before line 323 and $\hat{y}_{0,i}$ after it). Lines 308–310 and 361–371 run in every scenario; the bold lines run in v4 only. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.

```
176 def build_calib_design(mode, m_tr, m_va, mb_tr, mb_va, extra_tr, extra_va, y_tr, patm
        =760.0, rq=0.8):
177 ··· lines 177-184 omitted:  the docstring ···
185     m_tr = np.asarray(m_tr, np.float64); m_va = np.asarray(m_va, np.float64)
186     cols_tr, cols_va, names = [np.ones(len(m_tr)), m_tr], [np.ones(len(m_va)), m_va], ["
        c0", "w_persistence"]
187     ex_tr = extra_tr or {}; ex_va = extra_va or {}
188
189     def fetch(name, fill_tr, fill_va):
190         a = np.asarray(ex_tr.get(name, np.full(len(m_tr), np.nan)), np.float64)
191         b = np.asarray(ex_va.get(name, np.full(len(m_va), np.nan)), np.float64)
192         a = np.where(np.isfinite(a), a, fill_tr); b = np.where(np.isfinite(b), b, fill_va)
193         return a, b
194
195     if mode in ("calib_pt", "calib_pt_fio2", "calib_pt_asym"):
196         a, b = fetch("runmean_SaO2m", m_tr, m_va)              # no history yet -> the
        anchor itself
197         cols_tr.append(a); cols_va.append(b); names.append("w_patient_mean")
198 ··· lines 198-224 omitted:  the other anchor modes of the code base, used by no scenario
        of this paper ···
225     return np.column_stack(cols_tr), np.column_stack(cols_va), names
226 ··· lines 226-307 omitted:  other functions, and the start of the function _predict_fold
        ···
308         mass_balance = str(params.pop("mass_balance", "off")).lower()
309         CALIB_MODES = ("calib", "calib_asym", "calib_pt", "calib_pt_asym", "calib_fio2",
        "calib_pt_fio2")
310         if mass_balance in CALIB_MODES and raw_base_train is not None:
311 ··· lines 311-316 omitted:  comment lines ···
317             A_tr, A_va, names = build_calib_design(mass_balance, raw_base_train,
        raw_base_val, raw_mb_train, raw_mb_val,
318                                                    anchor_extra_train, anchor_extra_val,
        y_training,
319                                                    patm=float(getattr(DataConfig, "
        patm_mmhg", 760.0)))
320             sq = np.sqrt(np.asarray(sw, dtype=np.float64))
321             coef, *_ = np.linalg.lstsq(A_tr * sq[:, None], y_training.astype(np.float64)
        * sq, rcond=None)
322             persist_va = raw_base_val.astype(np.float64)
323             raw_base_train = (A_tr @ coef).astype(np.float32)
324             raw_base_val = (A_va @ coef).astype(np.float32)
325 ··· lines 325-360 omitted:  logging, and an anchor mode used by no scenario of this paper
        ···
361         dtrain = xgb.DMatrix(
362             X_training,
363             label=y_training,
364             weight=sw,
365             base_margin=raw_base_train if raw_base_train is not None else None,
366         )
367         dval = xgb.DMatrix(
368             X_validation,
369             label=y_validation,
370             base_margin=raw_base_val if raw_base_val is not None else None,
371         )
```

2.5. The Boosted Tree Ensemble Shared by the Three Scenarios

The three scenarios share one boosted ensemble; what differs between them enters only through the initial prediction $\hat{y}_{0,i}$, and the final prediction of Section 2.6. This subsection therefore describes the ensemble once. The ensemble is XGBoost 2.1.4 unchanged in all three scenarios; Section 2.5.4 states the six relations of its algorithm that our code touches and the three places where it does, without reproducing their derivations, which are those of Friedman [61] and of Chen and Guestrin [62]. Our contribution enters the ensemble at one place: the initial prediction $\hat{y}_{0,i}$ from which the recursion of Equation (28) starts, which is $y(\tau_i)$ in X0 and XP, Equation (20), and **the patient-level anchor of Equation (21) in v4**. The final prediction of XP and v4, Equation (34), is formed after the last tree and is not part of the recursion. Units: y_i, $\hat{y}_{b,i}$, f_b, the leaf scores $s_{b,j}$ and the gradients $g_{b,i}$ are in %; w_i, the Hessians $h_{b,i}$ and their sums, λ and η, are dimensionless; α is in %, the unit of the gradient sums; γ and the split gain are in %^{2}; the inputs $x_{i,m}$ are Z-scores. The library that we patched and ran is XGBoost release 2.1.4; every line number of XGBoost cited in this paper refers to the release 2.1.4 files of that library, and the lines of our own additions are cited against the same sources (Appendix A).

Boosting algorithms, from the foundational AdaBoost [63] on, build ensemble models sequentially, each new tree being trained to correct the errors of its predecessors [62]. XGBoost [62] adds a sparsity-aware split-finding algorithm and a regularised learning objective, an approach since validated across many tabular clinical-prediction tasks [35,38]. The fixed counts are written as n with a subscript naming what is counted: $i \in \{1, \ldots, n_i\}$ indexes the training samples, $m \in \{1, \ldots, n_m\}$ the features, and $b \in \{1, \ldots, n_b\}$ the boosting rounds (or, equivalently, the trees). The one count that varies from tree to tree, the number of leaves, is written T_b, with leaf index $j \in \{1, \ldots, T_b\}$.

2.5.1. Input, Target, and Prediction

Each sample i is one mechanically ventilated patient at one hour, described by the feature vector $\mathbf{x}_i \in \mathbb{R}^{n_m}$ of Tables 1 and 2. The value of n_m is a property of the scenario (Section 2.2): in X0 and XP, the trees receive $n_m = 55$ columns, the six real-time signals at hour t, their 44 lagged copies, and five target-derived or static descriptors; in v4, they receive $n_m = 19$, five signals at t with two lags each and four descriptors. Grouping the columns by block, the input vector is

$$\mathbf{x}_i = \Big(\underbrace{x_{i,1}, \ldots, x_{i,n_s}}_{\substack{n_s \text{ signals} \\ \text{at hour } t}}, \underbrace{x_{i,n_s+1}, \ldots, x_{i,n_s+n_\ell}}_{\substack{\text{the same signals} \\ \text{at their lag hours}}}, \underbrace{x_{i,n_s+n_\ell+1}, \ldots, x_{i,n_m}}_{\substack{\text{target history and} \\ \text{static descriptors}}} \Big)^\top \in \mathbb{R}^{n_m}, \tag{24}$$

with $(n_s, n_\ell, n_m) = (6, 44, 55)$ in X0 and XP and $(5, 10, 19)$ in v4, where each component $x_{i,m} \in \mathbb{R}$ is a single scalar feature and the superscript $\top$ denotes transposition. The ground truth $y_i \in [0, 100]$ is the registered saturation S_aO_2m in per cent at hour t (Section 2.1), and $n_i = 4017$ in every scenario. The model's prediction for sample i is a single scalar, its estimate of y_i obtained by mapping $\mathbf{x}_i$ through the ensemble.

2.5.2. From the Input to a Prediction

Chen and Guestrin [62] construct the map as a sum of n_b regression trees, each tree adding one correction to the running estimate:

$$\hat{y}_{n_b,i} = \phi(\mathbf{x}_i) = \hat{y}_{0,i} + \eta \sum_{b=1}^{n_b} f_b(\mathbf{x}_i), \quad f_b \in \mathcal{F}, \tag{25}$$

where $\hat{y}_{n_b,i}$ is the raw boosted estimate of S_aO_2m for sample i after the last round, ϕ denotes the full ensemble, $\hat{y}_{0,i}$ is the per-row initial prediction of Section 2.4 (Equation (20) in X0 and XP, **Equation** (21) **in v4**), η is the learning rate of Equation (28), n_b is the number of boosting rounds (capped at 2000 and chosen per fold by early stopping: on average, 800.8 trees in X0 and XP, 87.8 in v4), and f_b is the b-th regression tree, drawn from the space $\mathcal{F}$ of regression trees. In X0, the final prediction is $\tilde{y}_i = \hat{y}_{n_b,i}$; in XP and v4, it is the blend of Equation (34). A regression tree is a binary tree of yes/no questions of the form "is feature m of the sample below the threshold ϑ?" with one number stored at every leaf; two objects describe tree b completely, its routing $q_b : \mathbb{R}^{n_m} \to \{1, \ldots, T_b\}$, which gives the index of the leaf that a sample reaches, and its leaf-score vector $\mathbf{s}_b = (s_{b,1}, \ldots, s_{b,T_b})^\top$, so that

$$f_b(\mathbf{x}_i) = s_{b,j}, \qquad j = q_b(\mathbf{x}_i). \tag{26}$$

Both objects are learned from the data: the leaf scores through the closed form of Equation (32) below, and the split rules, which feature each node tests and at what threshold, through the greedy search that maximises the gain of Equation (33); nothing in a tree is specified by hand. The same routing fixes which *training* samples share a leaf, and the objective below sums over those per-leaf groups; q_b reappears in that role in Equation (30).

2.5.3. The Objective, with Our Weights in It

The trees are selected to minimise a training objective that balances the fit against the complexity of the trees. The data-fit term is the squared-error loss $\ell(y_i, \hat{y}_i) = \frac{1}{2}(y_i - \hat{y}_i)^2$ summed over the training samples (the factor $\frac{1}{2}$ makes the gradient of Equation (29) clean and does not change the minimiser), and the complexity term charges each tree for its leaves and for the size of its leaf scores. To make the predictions more sensitive to clinically critical low saturations, we insert the cost-sensitive sample weight w_i of Equation (3) into each loss term; this single substitution yields the weighted objective actually optimised in this study,

$$\mathcal{L}(f_1, \ldots, f_{n_b}) = \sum_{i=1}^{n_i} w_i \cdot \ell(y_i, \hat{y}_i) + \sum_{b=1}^{n_b} \Omega(f_b), \qquad \Omega(f_b) = \gamma\, T_b + \frac{1}{2}\lambda \sum_{j=1}^{T_b} s_{b,j}^2 + \alpha \sum_{j=1}^{T_b} |s_{b,j}|, \tag{27}$$

which is the regularised objective of Chen and Guestrin [62] with the weights of Section 2.3 in it; $\gamma \geq 0$ (`gamma`, 0 in our runs) is charged per leaf, and $\lambda \geq 0$ (`reg_lambda`) and $\alpha \geq 0$ (`reg_alpha`) are the L_2 and L_1 strengths on the leaf scores. The two summations run over different index sets, the samples and the trees, and $\Omega(f_b)$ differs from tree to tree because T_b and the scores depend on b. For the tuned values of k_w and ν, the weights are constant in effect (Section 3.5); what they still do to the trees through their common scale is stated in Section 2.3.

2.5.4. Training the Trees: Original XGBoost, and the Three Places Where Our Code Enters

The trees are trained by the gradient-boosting algorithm of XGBoost 2.1.4 exactly as released. We reproduce none of its derivations: the additive recursion is that of Friedman [61], and the second-order objective, the closed-form leaf score, and the split gain are those of Chen and Guestrin [62], to which we refer for their proofs. We state only the six relations of that algorithm that our code touches, in the notation of this paper, so that each of our interventions can be located (Figure 2):

$$\hat{y}_{b,i} = \hat{y}_{b-1,i} + \eta\, f_b(\mathbf{x}_i), \qquad b = 1, \ldots, n_b, \tag{28}$$

$$g_{b,i} = w_i\,(\hat{y}_{b-1,i} - y_i), \qquad h_{b,i} = w_i, \tag{29}$$

$$\mathcal{I}_{b,j} = \{\, i : q_b(\mathbf{x}_i) = j \,\}, \tag{30}$$

$$G_{b,j} = \sum_{i \in \mathcal{I}_{b,j}} g_{b,i}, \qquad H_{b,j} = \sum_{i \in \mathcal{I}_{b,j}} h_{b,i}, \tag{31}$$

$$s^*_{b,j} = -\frac{T_\alpha(G_{b,j})}{H_{b,j} + \lambda}, \qquad T_\alpha(G) = \operatorname{sign}(G)\, \max(|G| - \alpha,\, 0), \tag{32}$$

$$\text{Gain} = \frac{1}{2}\left[\frac{T_\alpha(G_{b,L})^2}{H_{b,L} + \lambda} + \frac{T_\alpha(G_{b,R})^2}{H_{b,R} + \lambda} - \frac{T_\alpha(G_{b,L} + G_{b,R})^2}{H_{b,L} + H_{b,R} + \lambda}\right] - \gamma, \tag{33}$$

with $\mathcal{I}_{b,j}$ the set of training rows in leaf j, $s^*_{b,j}$ in %, G and α in %, H and λ dimensionless, and the gain and γ in %^{2}. In words, there is one sentence per relation. Equation (28) is the additive recursion: each round adds one regression tree to the running prediction, scaled by the learning rate η, starting from the anchor $\hat{y}_{0,i}$ of Equation (20) or (21); it runs on the training rows and, for early stopping, on the validation rows. Equation (29) gives the gradient and the Hessian of the weighted squared loss $\frac{1}{2}\, w_i\,(y_i - \hat{y}_{b-1,i})^2$ of a training row at the previous prediction; these two numbers per row are all that the tree of round b is fitted to, so the weights w_i and the anchor enter the trees here and only here, and because the recursion starts from $\hat{y}_{0,i}$ the first gradients are $w_i(\hat{y}_{0,i} - y_i)$: the trees learn the residual of the anchor, not y_i itself. Equation (30) is the set of training rows that the routing q_b of the tree sends to leaf j. Equation (31) sums the gradients and the Hessians over that set; $G_{b,j}$ and $H_{b,j}$ are the only statistics of the data that a leaf sees. Equation (32) is the closed-form leaf score that minimises the second-order expansion of the objective (27) over the rows of the leaf, with the L_1 soft threshold T_α in the numerator and the L_2 penalty λ in the denominator. Equation (33) is the reduction in that objective obtained by splitting a node into a left and a right child, less the per-leaf penalty γ; the split search evaluates it for every candidate threshold of every feature and keeps the largest. A node is split if the gain is positive, both children have a summed Hessian of at least `min_child_weight`, and the depth is below `max_depth`; otherwise, it becomes a leaf with the score of Equation (32). In the library, Equation (29) is `src/objective/regression_obj.cu`, Equation (32) is `CalcWeight` in `src/tree/param.h`, Equation (33) is `CalcGain` applied inside `EnumerateSplit` in `src/tree/hist/evaluate\_splits.h`, and the scaling by η of Equation (28) is applied where the leaf scores are written into the tree; the threshold chosen for a node is the variable `split_pt` that `EnumerateSplit` selects, and it can be read back from the fitted booster with `trees_to_dataframe()` (the `Split` column) or `get_dump()`.

Our code enters this algorithm at three places, all of them before the first tree is grown, and modifies nothing inside it. First, the starting value $\hat{y}_{0,i}$ of Equation (28) is our anchor, the last available reading in X0 and XP, Equation (20), and **the patient-level anchor of Equation** (21) **in v4**, passed as the `base_margin` of both matrices (`utils_prediction.py`, Listing 1) and copied by the library into the prediction buffer of every row before the first tree (`Predictor::InitOutPredictions`, `src/predictor/predictor.cc`); consequently, every gradient of Equation (29) is evaluated relative to the anchor, and every leaf score of Equation (32) is a correction to it rather than to a constant. Second, the weights w_i of Equation (3), our relative-entropy criterion of Section 2.3, are the `weight` of the training matrix and multiply every gradient and Hessian (`regression_obj.cu`). Third, the hyperparameters η, λ, α, γ, `max_depth`, `min_child_weight`, the two subsampling ratios, and the early-stopping patience are set per scenario. The library's evaluation of the validation rows after each round (`src/learner.cc`) sees the unblended $\hat{y}_{b,i}$ and chooses the number of trees n_b by early stopping (Section 2.8.7): on average, 800.8 trees per fold with

12.0 leaves each in X0 and XP, and 87.8 trees with 5.4 leaves in v4. After the last tree, the final prediction of Section 2.6 is formed; it is ours and is not part of the training. The reason the anchor changes what the trees learn is visible in Equation (29): with the raw persistence anchor, the first gradients are the errors of a noisy nine-hour-old value, and the trees spend their capacity on pulling it toward the patient's level, which appears as a negative variance share of the trees in Section 3.3; with the patient-level anchor, that work is done in closed form before boosting, and a far smaller ensemble suffices.

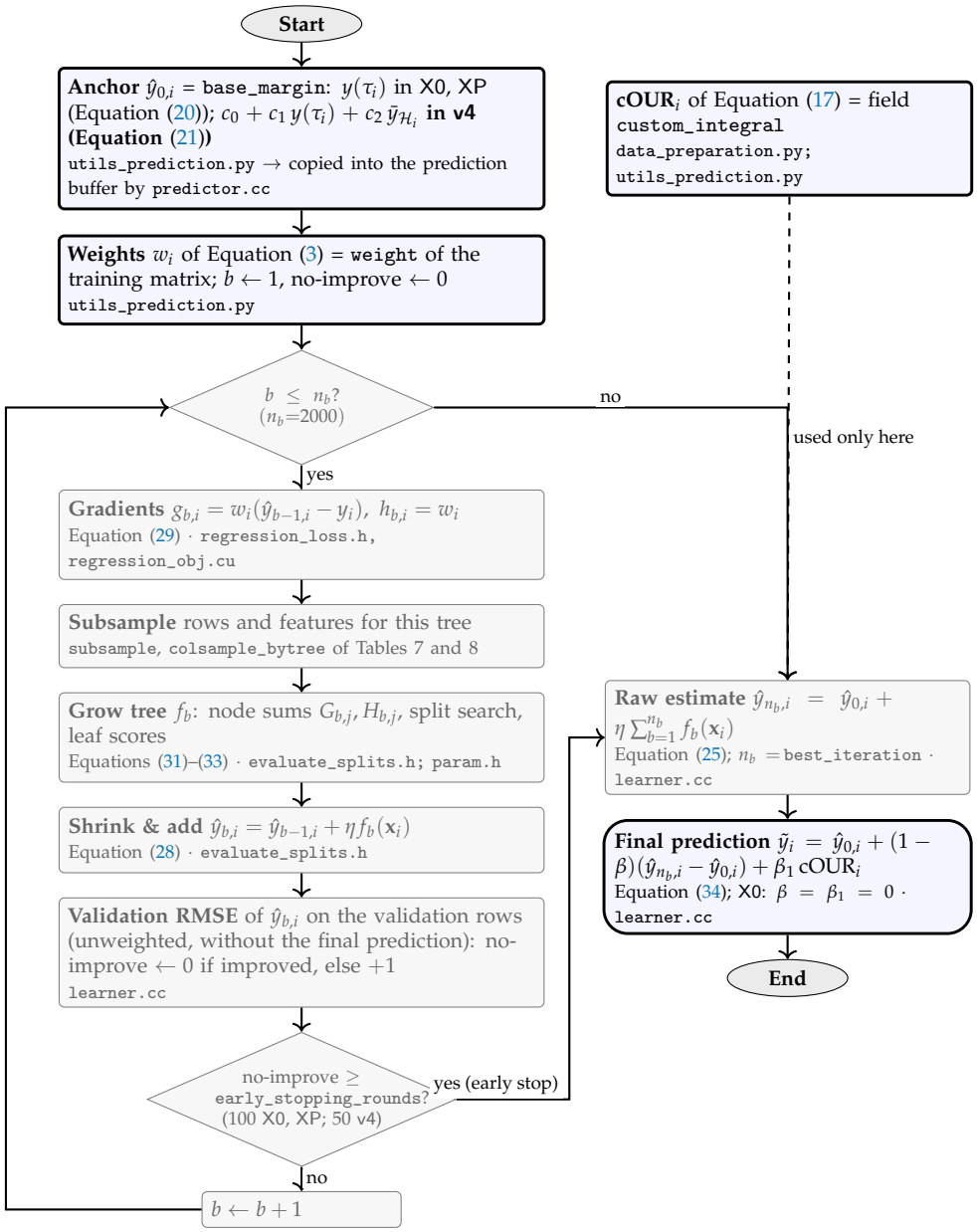


Figure 2. The boosting loop and where our code enters it (UML activity diagram). Grey blocks are original XGBoost 2.1.4 [62]; blocks with a black frame are ours, present in all three scenarios; the bold line of the *Anchor* block is v4 only. Our code sets the starting value $\hat{y}_{0,i}$ (the anchor), the weights w_i, and the cumulative oxygen uptake rate cOUR$_i$ before the loop and forms the final prediction after it; the loop, the tree recursion, and the early stopping are unchanged. Every block cites its equation and the source functions of release 2.1.4.

2.6. The Final Prediction

The trees give the raw boosted estimate $\hat{y}_{n_b,i}$ of Equation (25). After the last tree, our C++ function `ApplyAnchorBlend` forms the final prediction $\tilde{y}_i$ (%) of every row, training and validation rows alike:

$$\tilde{y}_i = \hat{y}_{0,i} + (1-\beta)\left(\hat{y}_{n_b,i} - \hat{y}_{0,i}\right) + \beta_1\,\mathrm{cOUR}_i \qquad [\mathsf{XP}] \tag{34}$$

where the dimensionless $\beta \in [0,1]$ (`custom_b`) pulls the prediction toward its initial value and β_1 (% kg h^{-1} a.u.$^{-1}$; `custom_b1`) weighs the cumulative oxygen uptake rate of Equation (17): $(\beta,\beta_1) = (0,0)$ in X0, $(0.1418, 0.00139)$ in XP (`config.py`), and $(0.3677, 0.00124)$ in v4 (locked configuration file). Substituting Equation (25), which gives $\hat{y}_{n_b,i} - \hat{y}_{0,i} = \eta \sum_b f_b(\mathbf{x}_i)$, together with Equations (21) and (17), into Equation (34) writes the prediction of v4 in its original quantities:

$$\tilde{y}_i = \underbrace{c_0 + c_1\,y(\tau_i) + c_2\,\bar{y}_{\mathcal{H}_i}}_{\hat{y}_{0,i}} + (1-\beta)\,\eta \sum_{b=1}^{n_b} f_b(\mathbf{x}_i) + \beta_1 \frac{D}{m_i^{kg}} \frac{1}{|\mathcal{L}_i|} \sum_{\ell \in \mathcal{L}_i} \mathrm{OUR}(t_i - \ell) \qquad [\mathsf{v4}] \tag{35}$$

In X0 and XP, the first term is $y(\tau_i)$, Equation (20). Table A2 in Appendix A gives, for every term of Equation (35), the Python variable that holds it, how it reaches `src/learner.cc`, where it is read, and when it is used; Listing 2 shows the C++ lines.

The predicted saturation is formed in `Predict`: `PredictRaw` evaluates the trees from $\hat{y}_{0,i}$ and gives $\hat{y}_{n_b,i}$, and `ApplyAnchorBlend` overwrites it in place with $\tilde{y}_i$. Python receives $\tilde{y}_i$ from `bst.predict(dval)` for the validation rows, which are the reported out-of-fold predictions, and from `bst.predict(dtrain)` for the training rows. β and β_1 therefore act on the predictions of training and validation rows alike, but not on the training loss or the early-stopping criterion: `UpdateOneIter` and `EvalOneIter` never call `ApplyAnchorBlend`. For $\beta \in [0,1]$ and $\beta_1 = 0$, Equation (34) is the convex combination $(1-\beta)\,\hat{y}_{n_b,i} + \beta\,\hat{y}_{0,i}$; hence, by the triangle inequality,

$$|\tilde{y}_i - y_i| \le (1-\beta)\,|\hat{y}_{n_b,i} - y_i| + \beta\,|\hat{y}_{0,i} - y_i| \le \max\left(|\hat{y}_{n_b,i} - y_i|,\ |\hat{y}_{0,i} - y_i|\right), \tag{36}$$

which is why `custom_b` is declared on $[0,1]$ and why Equation (34) is applied once, after the last tree: applied once per boosting round, it would compound to $(1-\beta)^{n_b}$ and Equation (36) would not hold.

That β and β_1 act only on the output is a design decision, and we state it in full because the words *physics-informed* might suggest that the two coefficients enter the training objective. They do not: the trees of XP and v4 are trained exactly as those of X0 would be with the same anchor (same gradients, same early stopping), and the two coefficients act once on the prediction. The alternative, placing Equation (34) inside the loss, would not give what the word shrinkage promises. With $\tilde{y}_i$ in the loss, the trees would minimise $\sum_i w_i\,[\,y_i - \hat{y}_{0,i} - \beta_1\,\mathrm{cOUR}_i - (1-\beta)\,\eta \sum_b f_b(\mathbf{x}_i)\,]^2 + \sum_b \Omega(f_b)$, which is the problem of Equation (27) for the rescaled residual $(y_i - \hat{y}_{0,i} - \beta_1\,\mathrm{cOUR}_i)/(1-\beta)$ with λ, α, γ, and the minimum child weight divided by $(1-\beta)^2$: at the optimum, the trees grow by the factor $1/(1-\beta)$ to undo the shrinkage, and β survives only as a change in regularisation. Applied once after the last tree, β keeps its meaning as a convex combination of the boosted estimate and the anchor, with the bound of Equation (36). The price is that β and β_1 are not fitted by the training loss but chosen, together with the other hyperparameters, on the out-of-fold record (Section 2.8.7); Section 3.4 measures what the two coefficients contribute and what choosing them on the scored folds is worth, by re-choosing them with each fold left out.

Listing 2: Scenarios XP and v4: the final prediction, Equation (34), in `src/learner.cc`: the declarations of β (`custom_b`) and β_1 (`custom_b1`), the single call from `Predict`, and `ApplyAnchorBlend`, in which `anchor[i]` is $\hat{y}_{0,i}$, `preds[i]` on entry is $\hat{y}_{n_b,i}$, `integral_val` is cOUR_i, and line 1504 is Equation (34). Line numbers are those of the patched file. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.]Scenarios XP and v4: the final prediction, Equation (34), in `src/learner.cc`: the declarations of β (`custom_b`) and β_1 (`custom_b1`), the single call from `Predict`, and `ApplyAnchorBlend`, in which `anchor[i]` is $\hat{y}_{0,i}$, `preds[i]` on entry is $\hat{y}_{n_b,i}$, `integral_val` is cOUR_i, and line 1504 is Equation (34). Line numbers are those of the patched file. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.

```
359     DMLC_DECLARE_FIELD(custom_b)
360         .set_default(0.0f)
361         .set_range(0.0f, 1.0f)
362         .describe(
363             "Anchor blend applied once to the output: y = (1-b)*y + b*base_margin. "
364             "Restricted to [0,1]: the output is then a convex combination of the boosted
     model "
365             "and the anchor and can never be worse than the worse of the two.");
··· lines 366-373 omitted:  two parameters used by no scenario of this paper ···
374     DMLC_DECLARE_FIELD(custom_b1)
375         .set_default(0.0f)
376         .set_range(-1.0f, 1.0f)
377 ··· lines 377-379 omitted:  the description string ···
380 ··· lines 380-1390 omitted:  unrelated code ···
1391     } else {
1392       auto& prediction = prediction_container_.Cache(data, ctx_.Device());
1393       this->PredictRaw(data.get(), &prediction, training, layer_begin, layer_end);
1394       // Copy the prediction cache to output prediction. out_preds comes from C API
1395       out_preds->SetDevice(ctx_.Device());
1396       out_preds->Resize(prediction.predictions.Size());
1397       out_preds->Copy(prediction.predictions);
1398
1399       // new line
1400       this->ApplyAnchorBlend(data.get(), out_preds);
1401
1402       if (!output_margin) {
1403         obj_->PredTransform(out_preds);
1404 ··· lines 1404-1475 omitted:  unrelated code ···
1476   void ApplyAnchorBlend(DMatrix* data, HostDeviceVector<bst_float>* io_preds) const {
1477     float const b = tparam_.custom_b;
1478     float const b1 = tparam_.custom_b1;
1479 ··· lines 1479-1480 omitted:  two parameters used by no scenario of this paper ···
1481
1482     if (data->Info().base_margin_.Size() == 0) {
1483       return;
1484     }
1485
1486     auto& preds = io_preds->HostVector();
1487     auto const& anchor = data->Info().base_margin_.Data()->HostVector();
1488     size_t n_rows = preds.size();
1489
1490     // Check if Python passed our custom integral
1491     bool has_integral = (data->Info().custom_integral_.Size() == n_rows);
1492     auto const* integral_ptr =
1493         has_integral ? data->Info().custom_integral_.HostVector().data() : nullptr;
1494
1495 #pragma omp parallel for
1496     for (omp_ulong i = 0; i < static_cast<omp_ulong>(n_rows); ++i) {
1497       float anchor_val = anchor[i];
1498       float integral_val = has_integral ? integral_ptr[i] : 0.0f;
1499
```

```
··· lines 1500-1501 omitted:  comment lines ···
      float const d = preds[i] - anchor_val;
··· lines 1503-1503 omitted:  with the defaults used in every scenario of this paper the
    factor shrink equals 1.0f - b ···
      preds[i] = anchor_val + shrink * d + b1 * integral_val;
    }
  }
```

2.7. Anatomy of a Prediction: How Much Is Physics and How Much Is Trees?

Equation (35) splits every final prediction exactly into three parts, written next to every out-of-fold row by the pipeline (`anchor_part`, `tree_part`, `integral_part`; `utils_prediction.py`):

$$\tilde{y}_i = A_i + B_i + C_i, \qquad A_i = \hat{y}_{0,i}, \quad B_i = (1-\beta)\,\eta \sum_{b=1}^{n_b} f_b(\mathbf{x}_i), \quad C_i = \beta_1\,\mathrm{cOUR}_i, \tag{37}$$

all in %. For each part $X \in \{A, B, C\}$, the variance share and the work share are

$$\text{variance share}(X) = \frac{\mathrm{Cov}(X, \tilde{y})}{\mathrm{Var}(\tilde{y})}, \qquad \text{work share}(X) = \frac{\overline{|X - X^0|}}{\sum_{X'} \overline{|X' - X'^0|}}, \tag{38}$$

with $X^0 = y(\tau_i)$ for A and $X^0 = 0$ for B and C, and the error credit of the anchor is

$$\text{error credit} = \frac{\mathrm{MAPE}(y(\tau)) - \mathrm{MAPE}(\hat{y}_0)}{\mathrm{MAPE}(y(\tau)) - \mathrm{MAPE}(\tilde{y})}, \tag{39}$$

with MAPE as in Equation (40). By bilinearity of the covariance, substituting Equation (37) into Equation (38) gives $\mathrm{Cov}(A, \tilde{y}) + \mathrm{Cov}(B, \tilde{y}) + \mathrm{Cov}(C, \tilde{y}) = \mathrm{Var}(\tilde{y})$, so the three variance shares sum to 100%; a negative share means that the part moves against the prediction. The work share describes how far each part moves the prediction away from the last available measurement, and the error credit describes which fraction of the improvement over it the anchor earns before any tree is grown.

2.8. Training and Evaluation Strategy

To avoid overly optimistic performance estimates caused by temporal correlation within the same patient trajectory, we implemented a 5-fold grouped cross-validation strategy. In this configuration, data is partitioned such that all samples of a patient are kept together within a single fold. This ensures that no data from the same patient used during the training phase is ever present in the test set, effectively preventing data leakage. The patient-level fold assignment is shuffled with a fixed random seed (`random_state` = 42), which also seeds the XGBoost estimator, so that the entire pipeline is reproducible; every scenario uses the same folds.

2.8.1. Target-History Leakage and the Strict Gap

As two of the inputs are the target's own past values (S_aO_2m at $t-9$ and $t-10$), the validity of the reported error depends critically on *which* target lags are admitted as features. The sensor is defined for a nine-hour absence of the reading (Section 2.1): the freshest value of the target it may use is its 9-h lag, and any value of S_aO_2m from the intervening eight hours ($t-1, \ldots, t-8$), or any feature derived from *future* samples, lies outside that definition and would not exist in the situation that the sensor simulates. If such a near-past or future value of the target were allowed into the feature set, the model would in effect be shown a lightly time-shifted copy of the answer: the reported MAPE would collapse to an optimistically small figure, and the "virtual sensor" would be quietly

reading the very signal it is meant to reconstruct rather than inferring it from ventilator and gas measurements. The pipeline constructs only target lags of nine hours or more. It does not, however, discard the rows that lack a nine-hour lag: in the first nine hours of a stay, it fills the lag with the patient's first measurement (`fallback_to_first=True`), and `Sa02m_first` and the patient mean carry the same value, so these 287 rows (7.1%) see a value fresher than $t-9$, and 33 of them the current target itself. By the rule above, these rows are invalid; all results of Section 3 are computed without them, so that the estimator at hour t sees the real-time gas and ventilator channels plus target history no fresher than $t-9$. It is important to stress that the patient-grouped cross-validation above does *not* on its own guard against this particular failure: grouping by patient eliminates *between*-patient leakage, but a near-past or future copy of a patient's *own* target leaks *within* that patient's trajectory, which grouped folds cannot detect. The headline accuracy must therefore be read as conditional on the strict-gap construction; relaxing it, admitting sub-gap target lags, or back-filling missing values from later rows, would lower the apparent error without any gain in genuine predictive power, and any such configuration should be treated as invalid for the nine-hour-ahead claim.

The primary evaluation metric utilised in this study is the mean absolute percentage error (MAPE), as it provides a relative measure of error that is clinically informative across varying oxygenation levels. The MAPE is defined as

$$\mathrm{MAPE} = \frac{100\%}{n_i} \sum_{i=1}^{n_i} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{40}$$

Equation (40) is the pooled, row-weighted MAPE. Three patients (3, 18, and 21) hold about 85% of the squared error of every scenario, and 5 patients have hours with two readings, so the pooled MAPE mostly reports those patients. We therefore report, for every scenario, the per-patient MAPE

$$\mathrm{MAPE}_p = \frac{100\%}{n_{i,p}} \sum_{i:p(i)=p} \left| \frac{y_i - \tilde{y}_i}{y_i} \right| \tag{41}$$

and three summaries of it over the $n_p = 31$ patients: the *macro* MAPE, its mean; the *trimmed* MAPE, its mean without the three worst patients; and its *median*. The macro MAPE is the objective that the pruning and the tuning of v4 used. Tracking within a patient is measured by

$$R^2_{\mathrm{within},p} = 1 - \frac{\frac{1}{n_{i,p}} \sum_{i:p(i)=p} (y_i - \tilde{y}_i)^2}{\mathrm{Var}_p(y)}, \tag{42}$$

where $\mathrm{Var}_p(y)$ is the sample variance of the patient's registered values (divisor $n_{i,p} - 1$, as the pipeline computes it), which is positive only when the model beats the patient's own mean; its median over patients is reported together with the same quantity computed for persistence, the floor a virtual sensor must beat, and with the number of patients in which the model beats persistence. The pooled Pearson correlation r between $\tilde{y}$ and y is reported as in our earlier work, with the caveat that it is lifted by between-patient level differences.

2.8.2. Uncertainty: The Paired Patient Bootstrap

Every difference between two scenarios, and between a scenario and persistence, is tested by a paired bootstrap over patients. Let $\mathcal{P}$ be the $n_p^* = 30$ strict patients and $\mathcal{P}^{(b)}$, $b = 1, \ldots, B$, a resample of n_p^* patients drawn from $\mathcal{P}$ with replacement ($B = 2000$, seed 42); for each member of the pair, the summary M (pooled, macro, trimmed or median

MAPE, or r) is recomputed on $\mathcal{P}^{(b)}$ from the per-patient sums, so that a patient drawn twice counts twice, and the interval of the difference is

$$\Delta M = M_A - M_B, \qquad \mathrm{CI}_{95\%}(\Delta M) = \left[\, q_{0.025}\{\Delta M^{(b)}\},\ q_{0.975}\{\Delta M^{(b)}\} \,\right], \tag{43}$$

the 2.5th and 97.5th percentiles over the resamples. A difference is called resolved when its interval excludes zero. The same resampling gives the intervals of the areas under the curve defined next.

2.8.3. Decision-Level Evaluation

The MAPE measures the error of a number; a bedside use of the sensor asks whether the saturation at the end of the gap will be below a threshold k. We therefore also score every scenario as a detector of hypoxaemia at hour t, with the event $E_i = [\, y_i < k \,]$ for $k \in \{88, 90, 92, 94\}$% and the score $-\tilde{y}_i$, a lower prediction ranking higher. The area under the ROC curve is the Mann–Whitney probability that a random event row receives a lower prediction than a random non-event row [64],

$$\mathrm{AUC} = \frac{1}{n_1 n_0} \sum_{i:E_i=1} \sum_{j:E_j=0} \left[\mathbb{1}(\tilde{y}_i < \tilde{y}_j) + \tfrac{1}{2}\,\mathbb{1}(\tilde{y}_i = \tilde{y}_j) \right], \tag{44}$$

with n_1 event and n_0 non-event rows. Persistence is scored with $y(\tau_i)$ as its prediction. At the natural operating point, an alarm $A_i = [\, \tilde{y}_i < k \,]$, we report the sensitivity $\Pr(A \mid E)$, the specificity $\Pr(\bar{A} \mid \bar{E})$, and the number of alarms. Because a reading that was already below k nine hours earlier is no discovery, the same quantities are also computed on the *at-risk* rows, $y(\tau_i) \geq k$, where the event is a new desaturation; on these rows, persistence can by construction never alarm, so its sensitivity is zero and the comparison isolates what a model adds to the last reading. The bootstrap above gives the intervals of the AUC and of its differences between v4 and persistence and between v4 and X0. This evaluation characterises the sensor; it is not the validation of an alarm, because the events are concentrated in a few patients (Section 3.1).

2.8.4. Intervals of the Plotted Predictions

The predictions of individual patients are shown in Section 3.1 with a 95% interval around every prediction. For scenario s and patient p, the interval is the prediction plus the empirical 2.5th and 97.5th percentiles of the out-of-fold residuals $y_j - \tilde{y}_j$ of s over the strict rows of the *other* patients, truncated to the physical range,

$$\left[\, \tilde{y}_i + q_{0.025}\{y_j - \tilde{y}_j\}_{j \notin p},\ \ \tilde{y}_i + q_{0.975}\{y_j - \tilde{y}_j\}_{j \notin p} \,\right] \cap [0, 100], \tag{45}$$

so that the shown patient never contributes to its own band; the coverage of the bands over all strict rows, patient by patient left out, is reported with the figure.

2.8.5. The Age of the Last Reading

The nominal gap is $D = 9$ h, but the actual age $D_i = t_i - \tau_i$ of the reading that a row uses, Equation (18), exceeds D whenever no reading fell in the hour $t_i - D$. Section 3.6 reports the distribution of D_i and, for every scenario, the dependence of the absolute percentage error $e_i = 100\,|y_i - \tilde{y}_i|/y_i$ on it: the pooled MAPE in three bins of D_i and Spearman's rank correlation

$$\rho_s(D, e) = \frac{\mathrm{Cov}\big(\mathrm{rank}(D_i),\ \mathrm{rank}(e_i)\big)}{\sigma_{\mathrm{rank}(D)}\,\sigma_{\mathrm{rank}(e)}}, \tag{46}$$

pooled over the strict rows and, as the mean over patients, within each patient.

2.8.6. Dependence on the Length of the Gap

The gap D enters the pipeline in five places, all driven by one setting, `virtual_sensor_gap` in `config.py`, which the environment variable `XGB_GAP` overrides: the target lags $t - D$ and $t - D - 1$, the persistence value $y(\tau_i)$, the patient mean $\bar{y}_{\mathcal{H}_i}$, the width D of the cumulative uptake of Equation (17), and the strict rule, which keeps the rows at least D hours after the patient's first row. To report how the error grows with the gap, the three scenarios are retrained with the same folds, lag sets, hidden columns, and hyperparameters at $D = 4, 5, 6, 7, 8$, and 9 h and scored on the strict rows of each gap (Section 3.7); the OUR values are still sampled at the lag hours of Table 1, and their average is multiplied by the gap D of the run.

2.8.7. Training Rows and Validation Rows

In each fold, the rows of 24 or 25 patients are training rows, and the rows of the other 6 or 7 patients are validation rows; the reported predictions are those of the validation rows. Table 6 lists every place where the two are treated differently.

Table 6. What differs between training and validation rows within a fold. File and line refer to the delivered sources.

Step	Training Rows	Validation Rows	Code
standardisation of the tree inputs	scaler fitted and applied	training scaler applied	`data_preparation.py`
$y(\tau_i)$, $\bar{y}_{\mathcal{H}_i}$, cOUR_i	from the row's own past	the same formula	`data_preparation.py`
anchor coefficients, Equation (22)	fitted	applied, not fitted	`utils_prediction.py`
sample weights w_i	in the loss	none	`utils_prediction.py`
trees f_b	fitted to the gradients	not used for fitting	`utils_prediction.py`
$\hat{y}_{b,i}$ after each round	enters the gradients	enters the early-stopping RMSE, unweighted and without the blend	`learner.cc`
number of trees n_b	—	decided by early stopping	`utils_prediction.py`
final prediction, Equation (34)	computed	computed; reported out-of-fold	`learner.cc`

Two consequences follow. The training loss and the early-stopping criterion never see β and β_1: the blend of Equation (34) exists only at prediction time, by the design decision stated in Section 2.6. The validation rows choose n_b through early stopping, so they are not entirely unseen during training; the folds are identical in every scenario, so the comparison between scenarios is not affected.

2.8.8. Hyperparameters of Each Scenario

X0 and XP use the configured operating point of our pipeline (`config.py`; Table 7); X0 differs from XP only in $\beta = \beta_1 = 0$. The hyperparameters of v4 (Table 8) were selected by a tree-structured Parzen estimator search (Optuna [65]) on the same patient-grouped folds for the 19-input tree with an earlier anchor, and were kept unchanged when the anchor of Equation (21) was introduced; v4 itself was not re-tuned. Its lag lists (Table 1) were reduced from those of X0 and XP by lag-group ablation on the same folds; the column `O2diffxPower` is hidden from its trees because it feeds Equation (17), and `Svoris` was hidden by a knockout test on the same folds. The configuration is fixed in the locked configuration file (the paragraph *Reproducibility* of Section 2.8). All hyperparameters of v4,

β and β_1 included, and the configured values of XP were thus chosen on the out-of-fold record of these same folds, without an outer validation loop; the reported errors are those of a selected configuration and are optimistic by the selection [66,67], while the comparison between the scenarios, which share the folds, is not biased by it. For the two output coefficients, the optimism can be measured exactly because they do not change the trees: Section 3.4 re-chooses β and β_1 with each fold left out.

Table 7. Hyperparameters of X0 and XP (`config.py` and the `TrainConfig` block). X0 sets $\beta = \beta_1 = 0$; everything else is identical.

Parameter (Name in the Code)	Value	Role
`learning_rate` (η)	0.0150	step applied to each tree's correction, Equation (28)
`n_estimators` (n_b cap)	2000	ceiling on the tree count; early stopping selects 800.8 on average
`early_stopping_rounds`	100	rounds without validation improvement before halting
`max_depth`	4	maximum tree depth
`min_child_weight`	0.632	minimum summed Hessian per leaf (the `IsValid` guard)
`reg_lambda` (λ)	17.17	L_2 penalty on leaf scores, Equation (32)
`reg_alpha` (α)	0.328	L_1 penalty on leaf scores
`gamma` (γ)	0	minimum gain to accept a split
`subsample`	0.505	fraction of rows per tree
`colsample_bytree`	1.0	fraction of columns per tree
`tree_method`	`hist`	histogram split finding
`custom_b` (β)	0 (X0)/0.1418 (XP)	anchor blend, Equation (34)
`custom_b1` (β_1)	0 (X0)/0.00139 (XP)	integral weight, Equation (34)
`custom_power` (ν), `custom_kp` (k_w)	3.656, 0.148	cost weights, Equation (3); inert on this target range (Section 3.5)
`random_state`	42	fold shuffle and model seed

2.8.9. Reproducibility

Each scenario is fully specified by one configuration: X0 and XP by `config.py` with $\beta = \beta_1 = 0$ or with their configured values, and v4 by a locked JSON file that records its lag lists, the columns hidden from the trees, the anchor mode, and every hyperparameter; `XGB_LOCKED=<file> python train.py` retrains it and reproduces the metrics of Section 3 exactly (same folds, same seed). The three scenarios were evaluated in one run on identical folds, and the out-of-fold prediction of every row was stored together with its three parts of Equation (37). The tables and the figure of the results, the bootstrap intervals of Equation (43), the decision-level quantities of Equation (44), the intervals of Equation (45), the per-fold anchors, the correlations of the parts, the age analysis of Equation (46), the cross-fitted output coefficients, the horizon runs, and the gain importance of the tree inputs are computed from those stored predictions and the summary files of the run by the evaluation scripts of the pipeline, which also re-derive every number

printed in this manuscript from the stored predictions, the summary files, the configuration, and the locked file and compare it with the text, so that the printed values and the files cannot drift apart. The evaluation scripts of this revision were drafted with the assistance of generative AI tools, were run and checked, and are maintained by the authors (see the Acknowledgments).

Table 8. Hyperparameters of v4 (locked configuration file `locked_model_v4.json`, keys `xgb_params`, `train`, and `runtime`), selected by the search described in the text.

Parameter (Name in the Code)	Value	Role
`learning_rate` (η)	0.0986	step applied to each tree's correction
`n_estimators` (n_b cap)	2000	ceiling; early stopping selects 87.8 trees on average
`early_stopping_rounds`	50	patience
`max_depth`	3	maximum tree depth
`min_child_weight`	97.96	minimum summed Hessian per leaf
`reg_lambda` (λ)	59.61	L_2 penalty on leaf scores
`reg_alpha` (α)	11.19	L_1 penalty on leaf scores
`subsample`, `colsample_bytree`	0.975, 0.927	row and column fractions per tree
`custom_b` (β)	0.3677	anchor blend, Equation (34)
`custom_b1` (β_1)	0.00124	integral weight, Equation (34)
`mass_balance` = `calib_pt`	—	the anchor of Equation (21), fitted per fold
`tree_exclude`	`O2diffxPower`, `Svoris`	columns hidden from the trees
`custom_power` (ν), `custom_kp` (k_w)	9.174, 0.928	cost weights; inert (Section 3.5)

3. Results

All results are out-of-fold predictions under the patient-grouped five-fold cross-validation of Section 2.8, with identical folds for the three scenarios. Following the rule of Section 2.8, they are scored on the 3730 rows of 30 patients whose inputs contain no S_aO_2m value younger than nine hours (Section 3.7). A patient counts as beating persistence when its $R^2_{\text{within},p}$, Equation (42), exceeds that of $y(\tau_i)$.

3.1. Accuracy and Size of the Three Scenarios

Table 9 reports the accuracy and size, and Table 10 reports the paired patient bootstrap interval of every difference (Equation (43)). The C++ terms of XP lower the macro MAPE by 0.031 percentage points, with a 95% interval $[-0.066, -0.001]$, while the intervals of the pooled (-0.009, $[-0.055, +0.032]$), trimmed (-0.026, $[-0.059, +0.002]$), and median per-patient MAPE (-0.075, $[-0.204, +0.050]$) include 0; 18 of the 30 patients improve. The effect of XP is therefore small and resolved only for the mean per-patient error. The patient-level anchor of v4 lowers, relative to XP, the pooled MAPE by 0.156 ($[-0.283, -0.015]$), the macro MAPE by 0.167 ($[-0.281, -0.052]$), and the trimmed MAPE

by 0.138 ([−0.242, −0.022]), while the change of the median, −0.057 ([−0.307, +0.125]), is not resolved; 23 of the 30 patients improve. Against persistence, v4 lowers the pooled MAPE by 0.573 ([−0.866, −0.264]) and raises r by 0.064 ([+0.041, +0.113]), the only change in the correlation that is resolved. v4 raises the pooled correlation from 0.754 to 0.793 and the median within-patient R^2 from −0.398 to −0.159, beats persistence in 29 of the 30 patients, and does so with 19 inputs instead of 55, 88 trees instead of 801, and 474 leaves per fold instead of 9637. Keeping only the last of two readings in the same hour (3427 strict rows) changes the pooled MAPE to 3.912% (X0), 3.892% (XP), and 3.743% (v4) and leaves the ranking unchanged.

Figure 3 illustrates the forecasting behaviour for two distinct cases: Patient 13 and Patient 27. Both panels display the registered values, the persistence baseline, and the predictions of XP and v4, along with the 95% prediction intervals from Equation (45).

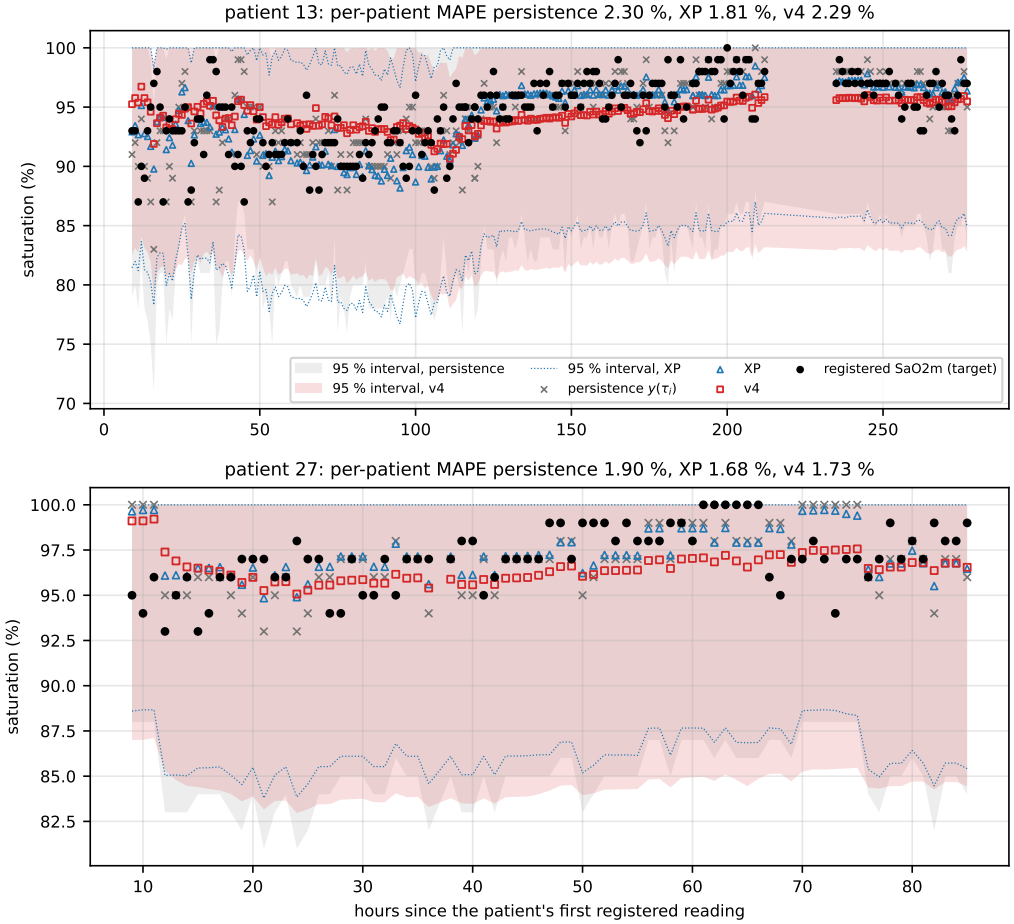


Figure 3. Two patients over their stays (strict rows; hours counted from the patient's first registered reading). (**Top**): Patient 13, showcasing a wider variation in registered values over a longer stay. While XP follows the delayed, hour-to-hour fluctuations, v4 damps these variations to indicate the patient's baseline trend. (**Bottom**): Patient 27, representing typical accuracy (the per-patient MAPE of v4 closest to the median among patients whose readings never fall below 85%). Both panels show the registered S_aO_2m (black), the persistence value $y(\tau_i)$ (grey crosses), XP (blue triangles), and v4 (red squares), along with the 95% prediction intervals of Equation (45) around persistence (grey band), v4 (red band), and XP (dotted blue lines).

Table 9. Out-of-fold accuracy and size on the 3730 rows of 30 patients whose inputs contain no S_aO_2m value younger than nine hours; identical patient-grouped folds. Trees and leaves are per-fold means. The persistence column scores $y(\tau_i)$ itself, with no model.

Quantity	Persistence	X0	XP	v4
tree inputs n_m	—	55	55	19
trees per fold	—	800.8	800.8	87.8
leaves per fold	—	9636.6	9636.6	474.4
pooled MAPE (%)	4.302	3.894	3.885	3.729
macro MAPE (%)	3.492	3.313	3.282	**3.115**
trimmed MAPE (%)	2.030	1.870	1.843	**1.705**
median per-patient MAPE (%)	1.962	1.874	1.799	**1.742**
pooled Pearson r	0.729	0.754	0.754	**0.793**
median R^2_{within}	−0.767	−0.415	−0.398	**−0.159**
patients beating persistence (of 30)	—	27	27	**29**

Patient 13 exhibits considerable variation while maintaining a steady course above 85%, with registered values ranging from 87 to 100% over the 248 strict rows of a 277-h stay. On this patient, the persistence baseline replays the wide fluctuations nine hours after they occur. XP closely follows this delayed replay, which allows it to track the hour-to-hour variations and yields a slightly lower MAPE for this specific patient (1.81% for XP compared to 2.29% for v4 and 2.30% for persistence). In contrast, v4 smooths out these delayed variations, damping the predictions toward the patient's underlying baseline. While it is less reactive to these transient changes than XP, v4 is marginally below persistence (2.29% against 2.30%) and provides a steadier indication of the patient's overall trend.

Patient 27 is of typical accuracy: among the patients whose registered values never fall below 85% on the strict rows (a steady course), it is the one whose per-patient MAPE of v4 is closest to the median (1.735%, median 1.742%). On patient 27, the persistence value replays the nine-hour-old reading with its excursions, XP follows it closely, and v4 holds the patient's level with a smaller spread; none of the three estimators anticipates the hour-to-hour changes in the registered value.

Finally, the 95% prediction intervals—derived from the out-of-fold residuals of the other patients—are wide, averaging about 15 percentage points for v4. This width is primarily driven by three patients with large errors that shape the residual distribution; leaving one patient out, they cover 93.1% (persistence), 92.8% (XP), and 92.4% (v4) of the strict rows.

Scored as detectors of a saturation below 90% at the end of the gap (Table 11; 643 of the 3730 strict rows, 11 patients, are below the threshold, and among the 3060 rows whose nine-hour-old reading was at or above it, 234 rows of 11 patients are new desaturations), persistence reaches an AUC of 0.900 [0.807, 0.963], X0 0.932, XP 0.931, and v4 0.939 [0.854, 0.983], with the anchor of v4 alone being 0.938. The difference v4 − persistence, +0.039 [+0.015, +0.077], is resolved; v4 − X0, +0.007 [−0.007, +0.014], is not. With the alarm rule "prediction below 90%", persistence finds 409 of the 643 event rows with 261 false alarms (sensitivity 0.636, specificity 0.915), v4 422 with 229 (0.656, 0.926), and the raw ensemble X0 more events at the price of more alarms (472 and 290; 0.734, 0.906). On the new desaturations, where persistence cannot alarm, v4 detects 71 of 234 (sensitivity 0.303) with 56 false alarms among the 2826 at-risk non-events (specificity 0.980; AUC 0.939), and X0 101 with 80 (0.432, 0.972; AUC 0.936). At the thresholds 88, 92, and 94%, the gain of v4 over persistence is 0.032 to 0.035 of the AUC and resolved at each, the gain over X0 is at most 0.005 and resolved at none, and on the new desaturations, the advantage of v4 over persistence grows from 0.067 at 88% to 0.102 at 94%, because a nine-hour-old reading says less and less about a crossing of a threshold that lies closer to the patient's usual level. The

ranking of the scenarios as detectors is the ranking of their MAPE, and the anchor alone detects as well as the full model; at 90%, the sensor anticipates fewer than a third of the new desaturations and the events come from 11 patients, so these numbers characterise a trend indicator, not an alarm.

Table 10. Paired patient bootstrap of the differences between the scenarios on the strict rows, Equation (43) (30 patients, 2000 resamples, seed 42): point estimate and 95% interval of the difference of each summary, first member minus second (percentage points of MAPE; units of r), and the number of patients whose per-patient MAPE is lower/higher in the first member. Intervals that exclude zero are in bold.

Summary	XP—X0	v4—XP	v4—X0	v4—Persistence
pooled MAPE	−0.009 [−0.055, +0.032]	**−0.156 [−0.283, −0.015]**	**−0.164 [−0.290, −0.042]**	**−0.573 [−0.866, −0.264]**
macro MAPE	**−0.031 [−0.066, −0.001]**	**−0.167 [−0.281, −0.052]**	**−0.198 [−0.328, −0.077]**	**−0.376 [−0.520, −0.224]**
trimmed MAPE	−0.026 [−0.059, +0.002]	**−0.138 [−0.242, −0.022]**	**−0.164 [−0.271, −0.053]**	**−0.325 [−0.486, −0.183]**
median MAPE	−0.075 [−0.204, +0.050]	−0.057 [−0.307, +0.125]	−0.132 [−0.335, +0.084]	−0.220 [−0.545, +0.098]
r	+0.000 [−0.010, +0.003]	+0.039 [−0.013, +0.064]	+0.039 [−0.022, +0.060]	**+0.064 [+0.041, +0.113]**
better/worse	18/12	23/7	22/8	26/4

Table 11. The scenarios as detectors of a saturation below 90 % at the end of the gap, Equation (44), on the 3730 strict rows (643 event rows of 11 patients) and on the 3060 at-risk rows whose nine-hour-old reading was at or above 90 % (234 new desaturations of 11 patients). AUC with its paired patient bootstrap interval, and, for the alarm rule "prediction below 90 %", sensitivity, specificity, and the number of alarms (true alarms in parentheses). Persistence uses $y(\tau_i)$ as its prediction and cannot alarm on an at-risk row.

Scenario	AUC [95% CI]	Sensitivity	Specificity	Alarms (True)
all strict rows: event $y_i < 90\,\%$				
persistence	0.900 [0.807, 0.963]	0.636	0.915	670 (409)
X0	0.932 [0.849, 0.980]	0.734	0.906	762 (472)
XP	0.931 [0.849, 0.979]	0.697	0.918	700 (448)
anchor of v4 alone	0.938 [0.860, 0.983]	0.703	0.915	713 (452)
v4	0.939 [0.854, 0.983]	0.656	0.926	651 (422)
at-risk rows, $y(\tau_i) \geq 90\,\%$*: event* $y_i < 90\,\%$				
persistence	0.851 [0.767, 0.886]	0.000	1.000	0 (0)
X0	0.936 [0.855, 0.961]	0.432	0.972	181 (101)
XP	0.932 [0.849, 0.959]	0.303	0.985	114 (71)
anchor of v4 alone	0.937 [0.869, 0.967]	0.316	0.979	132 (74)
v4	0.939 [0.855, 0.965]	0.303	0.980	127 (71)

3.2. The Anchor Coefficients of v4

Table 12 lists the coefficients of Equation (21) fitted on each training fold, with the standard error of their sum. c_1 and c_2 keep their sign in every fold, and their sum is 0.964 on average, so the anchor is, to four per cent, a weighted average that gives the nine-hour-

old reading 26% of the weight and the patient's own history 74%; the intercept is small, and changes sign across folds. The anchor alone, before any tree, has a pooled MAPE of 3.692% against 4.125% for the persistence value it recalibrates (all 4017 rows; 3.833% against 4.302% on the strict rows).

In fold 4, the sum exceeds one, $c_1 + c_2 = 1.028 \pm 0.013$, 2.2 standard errors above unity, and the intercept is negative, -2.86 ± 1.22. The two facts belong together. $y(\tau_i)$ and $\bar{y}_{\mathcal{H}_i}$ are strongly correlated (0.85 on the training rows of the fold), and both sit near the training level of 93.4%, so a slope sum above one is traded against a negative intercept along that level and the fitted map stays close to the other folds where the data lie: with both inputs at 90%, fold 4 returns 89.7%, and at 96%, it returns 95.8%, where the other four folds give 90.0 to 90.5% and 95.9 to 96.2%. The five anchors thus differ by less than one percentage point over the range that holds the data, and the fold 4 anchor is an expansion only in the sense that its fixed point, the level at which it returns its input, lies at 101.9% instead of between 91 and 100%. The weight of the nine-hour-old reading, $c_1 = 0.19$, is the smallest of the five folds, so the attenuation reading of Equation (23) holds in fold 4 as in the others; what differs is the level. Out-of-fold, on the 576 strict rows of its six validation patients, the fold 4 anchor alone (4.059%) is marginally worse than persistence (4.016%), the only fold in which this happens, while the full model reaches 3.577%; in the sample, on its 3395 training rows, the anchor improves persistence from 4.170 to 3.541%. Three coefficients fitted on 3395 rows do not overfit; the validation set of fold 4 holds one of the three patients that carry most of the squared error (patient 21), whose level the training patients do not represent, and the shortfall is a between-fold level shift that the trees, which see the ventilator channels, partly repair. We keep the unconstrained fit and report the sum with its standard error.

Table 12. Coefficients of the v4 anchor, Equation (21), per training fold, refitted from the stored out-of-fold rows (they reproduce the anchor stored with every row to within 7×10^{-6}), with the standard error of $c_1 + c_2$ from the least-squares fit.

Fold	c_0	c_1 (Last Reading)	c_2 (Patient History)	$c_1 + c_2$	SE of $c_1 + c_2$
1	9.325	0.2429	0.6585	0.901	0.017
2	2.267	0.2539	0.7213	0.975	0.013
3	5.645	0.3072	0.6361	0.943	0.011
4	−2.864	0.1941	0.8340	1.028	0.013
5	2.522	0.2419	0.7304	0.972	0.012
mean	3.379	0.2480	0.7161	0.964	

3.3. How Much of Each Prediction Is Physics

Table 13 applies the shares of Section 2.7 to the strict rows. In X0 and XP, the anchor is the raw persistence value and the trees' variance share is negative (−13.8%, −13.2%): the trees pull the noisy nine-hour-old value toward the patient's level, which is the shrinkage of Equation (23) done split by split with 801 trees. In v4, the anchor of Equation (21) does this in closed form: alone, it lowers the pooled MAPE from 4.302% (persistence) to 3.833%, which is 82% of the improvement of the full model (3.729%); it does 71% of the work, and the 88-tree ensemble corrects single rows while adding 1% of the variance. On the median patient, the trees add nothing: the anchor alone has a median per-patient MAPE of 1.703%; the full v4 model 1.742%. The identity $\tilde{y}_i = A_i + B_i + C_i$ holds to 4×10^{-6} on every row.

Which input matters most is answered by the same decomposition. Of the three parts, the anchor is the one that correlates with the target: on the strict rows, the Pearson correlation of A with y is 0.787 (its two inputs: the patient mean $\bar{y}_{\mathcal{H}_i}$ 0.786, the nine-hour-old reading $y(\tau_i)$ 0.729), that of the tree part B is 0.042, and that of the cumulative uptake

term C is -0.213; the last is a between-patient correlation (the correlation of the 30 patient means is -0.23; within patients, each patient's mean removed, it is $+0.01$), so the patients with the largest cumulative oxygen uptake per kilogram are not those with the highest saturation, and the term says nothing about the hourly changes of one patient. Within patients, the correlation of the anchor part with the target falls to 0.15 and that of the trees to -0.03, which is the quantitative form of the statement that the sensor reproduces levels rather than hourly changes. Among the inputs of the trees themselves, measured by the share of the total split gain averaged over the folds, the nine-hour-old reading dominates X0 and XP (`SaO2m_lag_9h` 37.2%, `SaO2m_first` 5.0%, and then O_2`xPress` at $t-12$ with 2.9%; the target-derived columns together 43%), whereas in v4, where the anchor has absorbed the level, the gain is spread over the elapsed treatment time (`count_hours` 13.2%), the nine-hour-old reading (10.4%), O_2`xPress` at t (9.0%), `GM102B_Volts` at $t-8$ (8.8%), and O_2`conc` at $t-6$ (6.9%), with the target-derived columns at 20%.

Table 13. Shares of Equations (38) and (39) in per cent on the strict rows. The variance shares of the three parts sum to 100.

Share (%)	X0	XP	v4
variance share of the anchor A	113.77	114.10	99.76
variance share of the trees B	−13.77	−13.22	1.21
variance share of the cumulative OUR term C	0.00	−0.88	−0.97
work share of the anchor A	0.00	0.00	70.59
work share of the trees B	100.00	84.75	21.26
work share of the cumulative OUR term C	0.00	15.25	8.16
error credit of the anchor	0.00	0.00	81.91
pooled MAPE of the anchor alone (%)	4.302	4.302	3.833

3.4. Ablation of the Two C++ Coefficients in v4

Table 14 evaluates the trained v4 model with β, β_1, or both set to zero in Equation (34). The two coefficients act only on the predictions (Section 2.6), so the trees are the same in every row of the table, and each row measures what a coefficient contributes to the predictions of the validation rows. The predictions with a coefficient set to zero are not stored row by row, so the table is on all 4017 rows. Setting $\beta = 0$ raises the pooled MAPE by 0.029 percentage points and the trimmed by 0.025, and changes the macro and the median by less than 0.003; setting $\beta_1 = 0$ raises the pooled MAPE by 0.002 and lowers the macro by 0.008, the trimmed by 0.007, and the median by 0.013. Both coefficients are kept by our rule that a physics coefficient is set to zero only when doing so changes the median per-patient MAPE by at most 0.001 and the macro by at most 0.005; the rule tests whether a coefficient matters, not the sign of its effect, and on all 4017 rows, the cumulative uptake term matters for the median and the macro error in the direction of a slightly lower error without it, while on the strict rows, its contribution together with β, the step from X0 to XP, is small and resolved only for the macro MAPE (Table 10).

As β and β_1 act on the output only (Section 2.6), what their selection on the scored folds is worth can be measured exactly, without retraining a tree. For each fold k, the pair (β, β_1) was re-chosen on the strict out-of-fold rows of the other four folds, minimising the macro MAPE with a penalty of 0.02 percentage points per patient below 30 that fails to beat persistence (a coarse grid followed by a fine one), and applied to the rows of fold k; the five held-out folds were then scored together. The re-chosen pairs range over $\beta = 0.00$ to 0.43 and $\beta_1 = 0.00018$ to 0.00107 around the locked values 0.3677 and 0.00124 (one fold prefers no shrinkage at all), and the cross-fitted record on the strict rows is a pooled MAPE of 3.752%, macro 3.131%, trimmed 1.718%, median 1.757%, $r = 0.790$, and 28 patients beating persistence, against 3.729%, 3.115%, 1.705%, 1.742%, 0.793, and 29 in

Table 9. The optimism of choosing the two coefficients on the scored folds is therefore 0.023 for pooled, 0.015 for macro, 0.013 for trimmed, 0.015 for median MAPE, and 0.003 for r, smaller than the resolved differences in Table 10, and cross-fitted v4 remains better than XP on every summary.

Table 14. Change in each metric when a coefficient of Equation (34) is set to zero in the trained v4 model; all 4017 rows, percentage points.

Set to Zero	Δ Pooled	Δ Macro	Δ Trimmed	Δ Median
$\beta_1 = 0$	+0.0022	−0.0076	−0.0066	−0.0134
$\beta = 0$	+0.0288	−0.0003	+0.0253	+0.0023
$\beta = \beta_1 = 0$	+0.0208	−0.0148	+0.0124	+0.0100

3.5. The Cost Weights Are Inert

The target ranges from 20 to 100% (Section 2.1). Over all 4017 rows, the largest relative deviation of w_i from k_w in Equation (3) is 1.0×10^{-4} in X0 and XP ($\nu = 3.656$, $k_w = 0.148$) and 9.0×10^{-14} in v4 ($\nu = 9.174$, $k_w = 0.928$), so the weights are constant to that precision, and the weighted least squares of Equation (22) and the weighted objectives of the trees are unweighted in effect: the tuned values sit at the variance end of the trade-off of Equation (6). The squared-relative-error end would matter only if $(1 - k_w)/|y|^\nu$ were comparable with k_w, which for y between 50 and 100% requires k_w of order 10^{-4} at $\nu = 2$, below the range that the tuning explored; exploring it is left for future work.

"Inert" holds for the relative weights, and hence for the anchor regression, which is invariant to a common factor. The common scale is not inert for the trees (Section 2.3): with $w_i = k_w$ on every row, the gradients and Hessians of Equation (29) are $k_w(\hat{y}_{b-1,i} - y_i)$ and k_w, so $G_{b,j} = k_w \sum \Delta$ and $H_{b,j} = k_w\, n_j$ in Equation (31), and the leaf score of Equation (32) becomes $-T_{\alpha/k_w}(\sum \Delta)/(n_j + \lambda/k_w)$: a constant weight k_w is exactly the unweighted model with λ/k_w, α/k_w and a minimum child weight of `min_child_weight`$/k_w$ rows. The trees of v4 ($k_w = 0.928$) therefore run at an effective $\lambda = 64.2$, $\alpha = 12.1$ and 105.5 rows per leaf, those of X0 and XP ($k_w = 0.148$) at $\lambda = 115.7$, $\alpha = 2.2$ and 4.3 rows per leaf; the early-stopping RMSE is scale-free, so the number of trees is unaffected. The tuned floors are thus part of the regularisation of the trees, which is one reason why the tuning of v4 returned a value of k_w so different from that of X0 and XP; the anchor and every physics term are untouched by it.

3.6. The Age of the Anchor and the Persistence Baseline

The persistence value is the most recent reading at or before $t_i - 9$ h, Equation (18). Of the 4017 rows, 89.5% have an anchor exactly nine hours old; 3.3% have an older one (median age 22.0 h, 90th percentile 74.8 h, maximum 81 h) because no reading fell in the preceding hour; and 7.1% lie in the first nine hours of a stay and use the patient's first reading, whose age is below nine hours and is zero in 33 rows (0.8%): the first row of each patient and, for patients 2 and 3, a second reading in that same hour. Because the target is integer-valued, the nine-hour-old value equals the target exactly in 17.1% of the rows; in the rows that have a reading at least nine hours old, this is quantisation of the label, not leakage, and it sets the floor that Table 9 reports for persistence.

Whether the age of the reading changes the error is answered in Table 15 and by Equation (46). On the strict rows, the actual age D_i is exactly nine hours in 3597 rows, between nine and 24 h in 79 rows of 15 patients, and above 24 h in 54 rows of 6 patients (median 9.0 h, 99th percentile 29.7 h, maximum 81 h). Spearman's correlation between D_i and the absolute percentage error is +0.039 for persistence, +0.006 for X0, +0.001 for XP and +0.024 for v4 pooled, and +0.016, −0.014, −0.014, and −0.001 as the mean within

patients: no scenario's error grows with the age of its reading. The older readings belong to few patients, and the errors of their bins reflect those patients' levels rather than the age (the rows above 24 h come from 6 patients and have the lowest MAPE of all for v4); the error of the sensor is governed by who the patient is, not by how old the last reading is, within the ages this cohort contains.

Table 15. Pooled MAPE (%) of persistence and of the three scenarios by the actual age D_i of the last available reading, strict rows, Equation (18).

Age of the Reading	Rows	Patients	Persistence	X0	XP	v4
exactly 9 h	3597	30	4.322	3.925	3.915	3.757
9 to 24 h	79	15	3.362	3.020	3.002	3.342
more than 24 h	54	6	4.363	3.083	3.133	2.449
all strict rows	3730	30	4.302	3.894	3.885	3.729

3.7. Strict Nine-Hour Check

The target-derived inputs use S_aO_2m values younger than nine hours in one situation: when a patient has no measurement at or before $t_i - D$, that is, in the first D hours of a stay. There, $y(\tau_i)$ and `Sa02m_lag_10h` fall back to the patient's first measurement (`data_preparation.py`, called with `fallback_to_first=True`), $\bar{y}_{\mathcal{H}_i}$ is filled with the same value (`utils_prediction.py`), and the tree input `Sa02m_first` is that first measurement too (`data_preparation.py`). This concerns 287 of the 4017 rows (7.1%) in every scenario, and in 33 of them, the value used is the current target itself; patient 14, with 6 registered hours spanning 5 h, has no row outside this window, so the strict rows cover 30 patients while the all-row figures cover all 31. No value from a later hour is used anywhere: the lags are taken with a backward merge, the patient mean stops at $t_i - D$, and `Sa02m_first` is the earliest reading. As the validation patients of a fold never appear among its training patients, these inputs in training rows carry no information about the validation rows; removing the 287 rows at scoring therefore leaves out-of-fold predictions whose inputs contain no S_aO_2m value younger than nine hours. Table 16 compares the two evaluations.

Table 16. Out-of-fold metrics on all 4017 rows (31 patients) and on the 3730 strict rows (30 patients).

Metric	X0		XP		v4	
	All	Strict	All	Strict	All	Strict
pooled MAPE (%)	3.785	3.894	3.761	3.885	3.603	3.729
macro MAPE (%)	3.267	3.313	3.205	3.282	3.019	3.115
trimmed MAPE (%)	1.956	1.870	1.901	1.843	1.746	1.705
median per-patient MAPE (%)	1.936	1.874	1.850	1.799	1.749	1.742
pooled Pearson r	0.763	0.754	0.764	0.754	0.801	0.793

The ranking of the three scenarios is the same in both evaluations. A deployable sensor under the strict rule starts nine hours after a patient's first reading; rows before that point would need a population prior as the anchor, which is not part of this study.

How the error grows with the length of the gap is shown in Table 17, in which the three scenarios are retrained at gaps of 4 to 9 h with the protocol of Section 2.8 and scored on the strict rows of each gap. From 4 to 9 h, the pooled MAPE of persistence grows from 4.001% to 4.302% and that of v4 from 3.487% to 3.729%, its macro MAPE from 2.897% to 3.115%, and its correlation falls from 0.816 to 0.793; the advantage of v4 over persistence lies between 0.506 and 0.606 percentage points of pooled MAPE at every gap, largest at 7 h, and v4 beats persistence in 27 to 29 of the patients with strict rows. The ranking v4 before XP before X0 holds at every gap, with XP and X0 tied at 7 h, and the size of the v4

ensemble stays between 76 and 99 trees per fold. The error of a nine-hour-old reading is thus 0.30 points above that of a four-hour-old one, and the sensor recovers about 19% of that growth.

Table 17. The three scenarios retrained at gaps D of 4 to 9 h (same folds, lag sets, hidden columns, and hyperparameters), scored on the strict rows of each gap (3887 to 3730 rows across 31 to 30 patients): pooled MAPE (%) of persistence and of the three scenarios, macro per-patient MAPE (%) and pooled r of v4, and the number of patients in which v4 beats persistence.

D (h)	Persistence Pooled	X0 Pooled	XP Pooled	v4 Pooled	v4 Macro	v4 r	v4 Beats Pers. (pts)
4	4.001	3.661	3.649	3.487	2.897	0.816	28 / 31
5	4.039	3.739	3.724	3.533	2.897	0.812	28 / 31
6	4.134	3.789	3.780	3.595	2.997	0.806	27 / 30
7	4.199	3.807	3.807	3.593	2.990	0.802	28 / 30
8	4.251	3.887	3.880	3.675	3.042	0.794	29 / 30
9	4.302	3.894	3.885	3.729	3.115	0.793	29 / 30

4. Discussion

Three findings carry the paper, all measured on the rows whose inputs contain no S_aO_2m value younger than nine hours. First, the physics pays off through the anchor. The patient-level recalibration of v4 is a two-coefficient linear term; yet, on its own, it earns 82% of the improvement over persistence, and with it, the pooled, macro, and trimmed errors fall by 0.14 to 0.17 percentage points relative to XP, with bootstrap intervals that exclude zero, while the tree ensemble shrinks to 19 inputs and 474 leaves. Second, the anatomy of Table 13 explains why: with a raw persistence anchor, the trees spend a negative variance share on shrinking a noisy measurement toward the patient's level, a task that Equation (23) performs in closed form once the patient's own history is available; freed from it, a much smaller ensemble corrects single rows. Third, the direct contribution of the gas signals through the physics terms is small; in particular, the cumulative oxygen uptake rate and the blend of XP lower the mean per-patient error by 0.031 percentage points, an effect whose interval only just excludes zero and which the pooled, trimmed, and median summaries do not resolve.

The detection analysis, the age of the reading, and the horizons add three readings. As a detector of a saturation below 90% at the end of the gap, v4 reaches an AUC of 0.939 against 0.900 for the last reading, a resolved difference, but only 0.007 above the raw ensemble; most of the detection, like most of the accuracy, is carried by the patient's level, and the anchor alone detects as well as the full model. On the hours in which the reading was still at or above 90% nine hours earlier, where the last reading cannot alarm at all, the sensor anticipates about three out of ten new desaturations at a false-alarm rate of 2% of the at-risk hours. The error does not grow with the age of the reading within the ages of this cohort (Section 3.6), but it grows with the length of the gap: from four to nine hours, the pooled error of the last reading rises by 0.30 points and that of v4 by 0.24, so the sensor's advantage over the reading widens with the gap (Section 3.7).

What the sensor does and does not do is now explicit. It reproduces each patient's level and beats the last reading in 29 of 30 patients; it does not resolve hourly changes for typical patients, whose hourly standard deviation of about 1.9 percentage points is at the label's integer quantisation, and the median within-patient R^2 stays negative in all scenarios. The median per-patient error is not resolved by either step, and on the median patient, the anchor of v4 alone is as accurate as the full model.

The claims are bounded by the design. The cohort is small (30 patients on the strict rows), and three patients hold most of the squared error, which is why the macro, trimmed, and median per-patient errors are reported next to the pooled one, and every difference

is tested with a paired patient bootstrap. The validation rows decide the number of trees through early stopping, which is common to all scenarios and therefore does not favour one of them. The training rows still include the first nine hours of the training patients' stays; this carries no information about the validation patients, but a pipeline that also drops these rows from training would be the natural next version. The claims are also bounded by the framing: the target is the monitor's pulse oximetry value registered once per hour by hand, not the saturation of a blood sample: it inherits the error of the oximeter, with its known dependence on perfusion and on skin pigmentation [7,10,11], and the transcription error of the chart, of which the 111% entry of Section 2.1 is an instance; the saturation rule of Equation (2) removes impossible values, not plausible wrong ones. The nine-hour gap is simulated: at the bedside, the monitor's reading is available, and what the sensor measures is the information that the gas and ventilator channels carry about the saturation nine hours ahead, which is the quantity that a trend indicator or a sampling aid needs. The hyperparameters—β and β_1 included—were selected on the out-of-fold record of the same folds; for the two output coefficients, the optimism was measured and is small (0.023 of the pooled MAPE, Section 3.4); for the tree hyperparameters, it was not measured, and the comparison between the scenarios, which share the folds, is unaffected by it.

Clinical Deployment

The output is a nine-hour-ahead estimate of the monitor's saturation computed from channels that are recorded in any case, the ventilator settings, and the expired-gas analysis; it would run beside the monitor, not in its place. Its constraints are the gas analysis device on the breathing circuit, the body mass, and nine hours of a patient's own registered history before it starts (Section 3.7). It applies to the ventilated patient whose procedures are settled and whose saturation is to be followed through the hours of occasional attention: a predicted level that drifts below the patient's own level, or below a threshold, is a reason to look at the patient before the next round, and Table 11 gives the operating characteristics of the simplest such rule as it stands, those of a screening aid and not of an alarm. It does not apply during the first nine hours of a stay, during a change in the ventilator circuit or of the device, or as a substitute for the monitor or the blood gas when a decision is due. Its longer-term uses are twofold: a check on the registered values, because a reading that departs from both the prediction and the patient's level by much more than the sensor's error is a candidate documentation error; and an aid to the timing of blood-gas sampling, each sample of which costs blood [68,69], because a sensor that holds over nine hours can support spacing the samples of a stable patient, a decision that remains clinical. The practical limits are those of the design: a single centre, 31 children, 1 ventilator model, and 1 gas analysis device, with no external validation, so that the model must be re-estimated for a new unit, with the anchor refitted at the least, and a prospective evaluation reported in accordance with the TRIPOD+AI statement [70] is required before any alerting use.

Within the scope of this Special Issue, the model is a multivariate time-series application with an explicit account of what each component contributes, and its loss is the relative-entropy criterion of our earlier estimators; a natural next step is to add information-theoretic descriptors of the ventilator channels (permutation or sample entropy over the same lag windows) as tree inputs and to test whether they carry the within-patient dynamics that the present inputs do not.

5. Conclusions

A virtual sensor for the arterial oxygen saturation registered from the bedside monitor, nine hours after the last available reading, was built as three nested scenarios of one code base and evaluated on identical folds on the rows whose inputs contain no saturation value younger than nine hours. Adding to XGBoost 2.1.4 a blend toward the anchor and the cumulative oxygen uptake rate over the gap, derived from an oxygen balance of the arterial blood (XP), produces a small improvement in the mean per-patient error. Replacing the raw nine-hour-old reading with a patient-level calibrated anchor (v4) lowers the pooled, macro, and trimmed errors with bootstrap intervals that exclude 0, beats the last reading in 29 of 30 patients, detects a saturation below 90% at the end of the gap with an area under the ROC curve of 0.939 against 0.900 for the last reading, and reduces the ensemble to 19 inputs and 474 leaves; the anchor alone earns 82% of the gain over persistence. The error does not depend on the age of the last reading within this cohort and, from a four- to a nine-hour gap, it grows by 0.24 points for the sensor against 0.30 for the reading alone. The two output coefficients act on the prediction only, by design, and choosing them on the scored folds costs 0.023 of the pooled MAPE when measured by cross-fitting. All formulas are matched line by line to the C++ and Python sources (Appendix A), and each scenario is reproducible from one configuration file.

Author Contributions: Conceptualisation, R.U.; methodology, R.U., S.R. and V.G. (Vytautas Galvanauskas); software, R.U. and N.A.P.R.; validation, R.U., N.A.P.R. and V.G. (Vytautas Galvanauskas); formal analysis, R.U. and N.A.P.R.; investigation, R.U. and N.A.P.R.; resources, R.U. and V.G. (Vytautas Galvanauskas); data curation, R.U. and T.K.; writing—original draft preparation, R.U. and N.A.P.R.; writing—review and editing, R.U., N.A.P.R. and V.G. (Vytautas Galvanauskas); visualisation, R.U. and N.A.P.R.; supervision, R.U., L.J. and V.G. (Vytautas Galvanauskas); project administration, R.U., V.G. (Vytautas Galvanauskas) and V.G. (Vaidotas Gurskis); funding acquisition, V.G. (Vytautas Galvanauskas). All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Research Council of Lithuania (Grant No. S-ITP-25-1).

Institutional Review Board Statement: The medical study to collect engineering data is carried out under permit BE-2-92 granted by the Kaunas Regional Biomedical Research Ethics Committee.

Informed Consent Statement: Informed consent was obtained from the legal representatives of all patients involved in the study.

Data Availability Statement: The changes to XGBoost 2.1.4 (`src/learner.cc`, and the field `custom_integral` in `include/xgboost/data.h`, `src/data/data.cc`, and `src/c_api/c_api.cc`) are available from the corresponding author upon reasonable request. The key functions of the Python pipeline are given in the listings of this paper. The patient data are not publicly available for privacy reasons.

Acknowledgments: The authors thank the Lithuanian University of Health Sciences Hospital Kauno Klinikos (Kaunas, Lithuania) for the collaboration in the pending clinical trials. During the preparation of this manuscript, the authors used generative AI tools (OpenAI ChatGPT 5.6 Sol, Google Gemini Pro, Anthropic Claude Opus 5.5, and Claude Fable 5.1) for language editing and for assistance in drafting and revising text, and Claude Fable 5.1 for drafting the analysis scripts that produced the tables and the figure for the revision (the paragraph *Reproducibility* of Section 2.8); the authors designed and ran all the implementations, checked all the results, reviewed and edited the content as needed, and take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

S_aO_2m	arterial oxygen saturation registered hourly from the bedside monitor's pulse oximeter, the target (data column `SaO2m`; the quantity written SpO_2 in the clinical literature)
SaO_2	arterial oxygen saturation of a blood sample (here, the ABL90 analyser; not used by the model)
AUC	area under the receiver operating characteristic (ROC) curve
CI	confidence interval (paired patient bootstrap)
cOUR	cumulative oxygen uptake rate over the gap per kilogram, Equation (17)
CPR	carbon dioxide production rate signal
ICU	intensive care unit
MAPE	mean absolute percentage error
OOF	out-of-fold
OUR	oxygen uptake rate
PEEP	positive end-expiratory pressure
SE	standard error
TPE	tree-structured Parzen estimator
WLS	weighted least squares

Appendix A. Code Listings and the Equation-to-Line Mapping

This appendix collects the code listings that Sections 2.1 and 2.4 refer to, with the marking convention of Section 2.2: Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only. Table A1 maps every equation of ours to the file, the function, and the lines that implement it; Table A2 follows each term of the final prediction, Equation (35), from the Python variable that holds it to the C++ line that uses it. The line numbers are those of the delivered sources; the lines of the original XGBoost 2.1.4 algorithm that our code touches are named in Section 2.5.4.

Listing A1: All scenarios: the saturation rule of Equation (2) in `load_data` (`data_preparation.py`), applied before any lag, running mean, or metric is formed; `_lo`, `_hi` are `target_valid_range` of `config.py` (0 and 100); the environment variable `XGB_TARGET_RANGE` overrides them (`off` disables the rule). The line numbers are those of the source file. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.

```
991 _rng = os.environ.get("XGB_TARGET_RANGE", "")
992 if _rng.lower() != "off" and target_column in data.columns:
993     try:
994         if _rng:
995             _lo, _hi = (float(x) for x in _rng.split(":"))
996         else:
997             from config import DataConfig as _DC
998             _lo, _hi = getattr(_DC, "target_valid_range", (0.0, 100.0))
999         _y = pd.to_numeric(data[target_column], errors="coerce")
1000        _bad = _y.notna() & ((_y < _lo) | (_y > _hi))
1001        if int(_bad.sum()):
1002            _sat = _y.clip(lower=_lo, upper=_hi)
1003            for _i in data.index[_bad]:
1004                logger.warning("target %s = %s outside [%g, %g] at patient %s, %s ->
    saturated to %g (documentation error)",
1005                               target_column, data.at[_i, target_column], _lo, _hi, data.
    at[_i, patient_id_column], data.at[_i, time_column], _sat.at[_i])
1006            data.loc[_bad, target_column] = _sat[_bad]
```

```
1007            logger.info("saturation validity rule: %d value(s) of %s outside [%g, %g]
                 saturated to the nearest bound", int(_bad.sum()), target_column, _lo, _hi)
1008        except Exception as _e:
1009            logger.warning("saturation validity rule not applied: %s", _e)
```

Listing A2: Scenarios XP and v4: the cumulative oxygen uptake rate over the gap, Equation (17): `compute_feature_window_integral` in `data_preparation.py`; `mean_rate` is the average over the lag set, and `gap_hours` is D. The line numbers are those of the source file. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.

```
18  def compute_feature_window_integral(df: pd.DataFrame, base_feature: str, gap_hours: int =
        9) -> pd.Series:
19      """Window integral = mean rate over the available window columns x gap_hours (per kg).
20      Scaled by the GAP (9 h), not by the number of lag columns, so pruning window lags
         changes the
21      estimate, not the units: with the current 9 columns (raw + 8 lags) the two are
         identical."""
22      window_cols = [c for c in df.columns if c == base_feature or c.startswith(f"{
         base_feature}_lag_")]
23      if not window_cols:
24          return pd.Series(0.0, index=df.index)
25
26      W = len(window_cols)
27      mean_rate = df[window_cols].mean(axis=1, skipna=True)
28      integral = mean_rate * float(gap_hours)
29  ··· lines 29-51 omitted:  the last known body mass of the patient (column Svoris; zeros
         treated as missing; the mean over the patients as the last resort) and the division
         of integral by it ···
52      return integral.fillna(0.0)
```

Listing A3: Scenarios XP and v4: the transfer of cOUR_i (`raw_int_train`, `raw_int_val`) to the library as the per-row field `custom_integral`, `utils_prediction.py`. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.

```
373         if raw_int_train is not None:
374             dtrain.set_float_info("custom_integral", raw_int_train)
375         if raw_int_val is not None:
376             dval.set_float_info("custom_integral", raw_int_val)
```

Listing A4: Scenario v4: the past mean $\bar{y}_{\mathcal{H}_i}$ of Equation (21): `compute_target_runmean` in `data_preparation.py`; `csum` and `cnt` are the running sum and count of the measurements, and `pos` the index of the last one at or before $t - D$. The column is registered as an anchor input and removed from the tree inputs by two lines of `config.py` (53–54): `anchor_extra_cols = ["O2conc", "CO2mmHg", "runmean_SaO2m"]` and `anchor_drop_from_tree = ["CO2mmHg", "runmean_SaO2m"]`. Grey lines: unmodified XGBoost 2.1.4 (all scenarios); black lines of regular weight: our code, active in XP and v4; **bold** lines: our code active in v4 only.

```
127 def compute_target_runmean(df: pd.DataFrame, target: str, gap_hours: int = 9,
128                           patient_column: str = "patient_id", time_column: str = "
        LocalTime") -> pd.Series:
129     """The patient's OWN running mean of the target over every measurement at or before t
         - gap_hours
130     (time-based, per patient). Legal at prediction time exactly like the persistence
         anchor, and the
131     natural target for shrinking a noisy blood gas: toward the patient's level, not the
         cohort's.
132     NaN until the patient has a first measurement >= gap_hours before t."""
133     out = pd.Series(np.nan, index=df.index)
134     if target not in df.columns or patient_column not in df.columns or time_column not in
         df.columns:
```

```
        return out
    for _, idx in df.groupby(patient_column).groups.items():
        g = df.loc[idx, [time_column, target]].sort_values(time_column)
        t = pd.DatetimeIndex(pd.to_datetime(g[time_column]).values); v = g[target].
 to_numpy(float)
        ok =  np.isnan(v)
        csum = np.cumsum(np.where(ok, v, 0.0)); cnt = np.cumsum(ok.astype(int))
        pos = np.searchsorted(t.values, (t - pd.Timedelta(hours=gap_hours)).values, side
 ="right") - 1
        safe = np.maximum(pos, 0)
        vals = np.where((pos >= 0) & (cnt[safe] > 0), csum[safe] / np.maximum(cnt[safe],
 1), np.nan)
        out.loc[g.index] = vals
    return out
```

Table A1. Where each of our equations is implemented: file, function, and line numbers in the delivered sources (line numbers of the frozen v4 code set; C++ lines of the patched `src/learner.cc`). The relations of the original algorithm, Equations (28)–(33), are in the release 2.1.4 files named in Section 2.5.4 and are unchanged.

Equation	What	File, Function	Lines (Listing)
Equation (2)	the registered value saturated to 0–100%	`data_preparation.py`, `load_data`; `config.py`, `target_valid_range`	Listing A1
Equation (3)	sample weights w_i	`utils_prediction.py`, `compute_sample_weights`; called in `_predict_fold`	228–231; 295
Equation (17)	cumulative uptake cOUR_i	`data_preparation.py`, `compute_feature_window_integral`	18–52 (Listing A2)
Equation (17), transfer	field `custom_integral`	`utils_prediction.py`, `_predict_fold`	373–376 (Listing A3)
Equation (18)	last available reading $y(\tau_i)$	`data_preparation.py`, `_time_based_lag_last_known`	195–241
Equation (21), $\bar{y}_{\mathcal{H}_i}$	the patient's past mean	`data_preparation.py`, `compute_target_runmean`; registered and hidden from the trees in `config.py`	127–145 (Listing A4); 53–54
Equations (21) and (22)	design matrix, weighted least squares, anchor of every row	`utils_prediction.py`, `build_calib_design`; `_predict_fold`	176–225; 308–324, with 321 Equation (22) and 323–324 Equation (21) (Listing 1)
Equation (20)/(21)	$\hat{y}_{0,i}$ handed to the library as `base_margin`	`utils_prediction.py`, `_predict_fold`; hidden columns in `_extract_base_margin`	361–371 (Listing 1); 28 ff.
Equation (34)	β, β_1 declared; the single call; `ApplyAnchorBlend`; the line	`src/learner.cc`	359–365, 374–376; 1400; 1476–1506; 1504 (Listing 2)
Equation (34), coefficients	β, β_1 of XP; of v4	`config.py`; locked configuration file	248–249; keys `custom_b`, `custom_b1`
Equation (3), coefficients	ν, k_w, ε of X0 and XP	`config.py`, class `TrainConfig`	329–331
Equations (43)–(46)	the tables and the figure of Section 3	the evaluation scripts of the pipeline, on the stored out-of-fold predictions	—

Table A2. The terms of Equation (35) in the code. File `utils_prediction.py` unless stated; C++ lines in `src/learner.cc`.

Term	Python	In C++	Used in
c_0, c_1, c_2	`coef[0..2];`	not present; they act through $\hat{y}_{0,i}$	through $\hat{y}_{0,i}$
$y(\tau_i)$, $\bar{y}_{\mathcal{H}_i}$	columns `SaO2m_lag_9h`, `runmean_SaO2m`	not present	through $\hat{y}_{0,i}$
$\hat{y}_{0,i}$	`raw_base_train`, `raw_base_val`; `base_margin`	`base_margin_`: start of the prediction buffer before the first tree; `anchor`	training rounds, early stopping, prediction
$\eta \sum_b f_b(\mathbf{x}_i)$	the trees, trained by `xgb.train`	`PredictRaw` (training), (early stopping),(prediction)	training rounds, early stopping, prediction
cOUR_i	`raw_int_train`, `raw_int_val`;`set_float_info("custom_integral", ...)`,	our field `custom_integral_` (declared in `include/xgboost/data.h`; filled by `MetaInfo::SetInfo`, `src/data/data.cc`; C API `src/c_api/c_api.cc`)	prediction
β, β_1	`params["custom_b"]`, `params["custom_b1"]`; `xgb.train(params=params, ...)`	`tparam_.custom_b`, `tparam_.custom_b1`	prediction

References

1. Dall'Ora, C.; Saville, C.; Rubbo, B.; Turner, L.; Jones, J.; Griffiths, P. Nurse staffing levels and patient outcomes: A systematic review of longitudinal studies. *Int. J. Nurs. Stud.* **2022**, *134*, 104311. https://doi.org/10.1016/j.ijnurstu.2022.104311.
2. Bertoni, M.; Spadaro, S.; Goligher, E.C. Monitoring Patient Respiratory Effort During Mechanical Ventilation: Lung and Diaphragm-Protective Ventilation. *Crit. Care* **2020**, *24*, 106. https://doi.org/10.1186/s13054-020-2777-y.
3. Nguyen, L.S.; Squara, P. Non-invasive monitoring of cardiac output in critical care medicine. *Front. Med.* **2017**, *4*, 200. https://doi.org/10.3389/fmed.2017.00200.
4. Athaya, T.; Choi, S. A Review of Noninvasive Methodologies to Estimate the Blood Pressure Waveform. *Sensors* **2022**, *22*, 3953. https://doi.org/10.3390/s22103953.
5. Dong, J.; Xu-Wilson, M.; Conroy, B.R.; Khemani, R.G.; Newth, C.J.L. Continuous noninvasive blood gas estimation in critically ill pediatric patients with respiratory failure. *Sci. Rep.* **2022**, *12*, 9853. https://doi.org/10.1038/s41598-022-13583-6.
6. Krzych, L.; Wojnarowicz, O.; Ignacy, P.; Dorniak, J. Be cautious during the interpretation of arterial blood gas analysis performed outside the intensive care unit. *Acta Biochim. Pol.* **2020**, *67*, 353–358. https://doi.org/10.18388/abp.2020_5178.
7. Jubran, A. Pulse oximetry. *Crit. Care* **2015**, *19*, 272. https://doi.org/10.1186/s13054-015-0984-8.
8. Chan, E.D.; Chan, M.M.; Chan, M.M. Pulse oximetry: Understanding its basic principles facilitates appreciation of its limitations. *Respir. Med.* **2013**, *107*, 789–799. https://doi.org/10.1016/j.rmed.2013.02.004.
9. Perkins, G.D.; McAuley, D.F.; Giles, S.; Routledge, H.; Gao, F. Do changes in pulse oximeter oxygen saturation predict equivalent changes in arterial oxygen saturation? *Crit. Care* **2003**, *7*, R67–R71. https://doi.org/10.1186/cc2339.
10. Sjoding, M.W.; Dickson, R.P.; Iwashyna, T.J.; Gay, S.E.; Valley, T.S. Racial Bias in Pulse Oximetry Measurement. *N. Engl. J. Med.* **2020**, *383*, 2477–2478. https://doi.org/10.1056/NEJMc2029240.
11. Wong, A.K.I.; Charpignon, M.; Kim, H.; Josef, C.; de Hond, A.A.H.; Fojas, J.J.; Tabaie, A.; Liu, X.; Mireles-Cabodevila, E.; Carvalho, L.; et al. Analysis of Discrepancies Between Pulse Oximetry and Arterial Oxygen Saturation Measurements by Race and Ethnicity and Association With Organ Dysfunction and Mortality. *JAMA Netw. Open* **2021**, *4*, e2131674. https://doi.org/10.1001/jamanetworkopen.2021.31674.
12. Agniel, D.; Kohane, I.S.; Weber, G.M. Biases in electronic health record data due to processes within the healthcare system: Retrospective observational study. *BMJ* **2018**, *361*, k1479. https://doi.org/10.1136/bmj.k1479.
13. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Sci. Rep.* **2018**, *8*, 6085. https://doi.org/10.1038/s41598-018-24271-9.
14. Sperrin, M.; Martin, G.P.; Sisk, R.; Peek, N. Missing data should be handled differently for prediction than for description or causal explanation. *J. Clin. Epidemiol.* **2020**, *125*, 183–187. https://doi.org/10.1016/j.jclinepi.2020.03.028.
15. van Smeden, M.; Groenwold, R.H.; Moons, K.G. A cautionary note on the use of the missing indicator method for handling missing data in prediction research. *J. Clin. Epidemiol.* **2020**, *125*, 188–190. https://doi.org/10.1016/j.jclinepi.2020.06.007.

16. Kadlec, P.; Gabrys, B.; Strandt, S. Data-driven Soft Sensors in the process industry. *Comput. Chem. Eng.* **2009**, *33*, 795–814. https://doi.org/10.1016/j.compchemeng.2008.12.012.
17. Patharkar, A.; Cai, F.; Al-Hindawi, F.; Wu, T. Predictive modeling of biomedical temporal data in healthcare applications: Review and future directions. *Front. Physiol.* **2024**, *15*, 1386760. https://doi.org/10.3389/fphys.2024.1386760.
18. Lee, J.M.; Hauskrecht, M. Modeling multivariate clinical event time-series with recurrent temporal mechanisms. *Artif. Intell. Med.* **2021**, *112*, 102021. https://doi.org/10.1016/j.artmed.2021.102021.
19. Dauvin, A.; Donado, C.; Bachtiger, P.; Huang, K.C.; Sauer, C.M.; Ramazzotti, D.; Bonvini, M.; Celi, L.A.; Douglas, M.J. Machine learning can accurately predict pre-admission baseline hemoglobin and creatinine in intensive care patients. *npj Digit. Med.* **2019**, *2*, 116. https://doi.org/10.1038/s41746-019-0192-z.
20. Kipnis, E.; Ramsingh, D.; Bhargava, M.; Dincer, E.; Cannesson, M.; Broccard, A.; Vallet, B.; Bendjelid, K.; Thibault, R. Monitoring in the Intensive Care. *Crit. Care Res. Pract.* **2012**, *2012*, 473507. https://doi.org/10.1155/2012/473507.
21. Qiu, X.; Tan, X.; Wang, C.; Chen, S.; Du, B.; Huang, J. A long short-term memory relation network for real-time prediction of patient-specific ventilator parameters. *Math. Biosci. Eng.* **2023**, *20*, 14756–14776. https://doi.org/10.3934/mbe.2023660.
22. The Acute Respiratory Distress Syndrome Network; Brower, R.G.; Matthay, M.A.; Morris, A.; Schoenfeld, D.; Thompson, B.T.; Wheeler, A. Ventilation with Lower Tidal Volumes as Compared with Traditional Tidal Volumes for Acute Lung Injury and the Acute Respiratory Distress Syndrome. *N. Engl. J. Med.* **2000**, *342*, 1301–1308. https://doi.org/10.1056/NEJM200005043421801.
23. Brower, R.G.; Lanken, P.N.; MacIntyre, N.; Matthay, M.A.; Morris, A.; Ancukiewicz, M.; Schoenfeld, D.; Thompson, B.T.; National Heart, Lung, and Blood Institute ARDS Clinical Trials Network. Higher versus Lower Positive End-Expiratory Pressures in Patients with the Acute Respiratory Distress Syndrome. *N. Engl. J. Med.* **2004**, *351*, 327–336. https://doi.org/10.1056/NEJMoa032193.
24. Zhang, L.; Ren, G.; Li, S.; Du, J.; Xu, D.; Li, Y. A novel soft sensor approach for industrial quality prediction based TCN with spatial and temporal attention. *Chemom. Intell. Lab. Syst.* **2025**, *257*, 105272. https://doi.org/10.1016/j.chemolab.2024.105272.
25. Wang, C.; He, T.; Zhou, H.; Zhang, Z.; Lee, C. Artificial intelligence enhanced sensors—Enabling technologies to next-generation healthcare and biomedical platform. *Bioelectron. Med.* **2023**, *9*, 17. https://doi.org/10.1186/s42234-023-00118-1.
26. Li, J.; Ma, J.; Omisore, O.M.; Liu, Y.; Tang, H.; Ao, P.; Yan, Y.; Wang, L.; Nie, Z. Noninvasive Blood Glucose Monitoring Using Spatiotemporal ECG and PPG Feature Fusion and Weight-Based Choquet Integral Multimodel Approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 14491–14505. https://doi.org/10.1109/TNNLS.2023.3279383.
27. Quamer, W.; Tseng, M.R.; Vyas, K.; Dave, D.; Villegas, C.; McKay, S.; DeSalvo, D.J.; Erranguntla, M.; Cote, G.; Gutierrez-Osuna, R. A multimodal physiological dataset for non-invasive blood glucose estimation. *Sci. Data* **2025**, *12*, 1822. https://doi.org/10.1038/s41597-025-06090-6.
28. Tsou, P.H.; Lin, Z.L.; Pan, Y.C.; Yang, H.C.; Chang, C.J.; Liang, S.K.; Wen, Y.F.; Chang, C.H.; Chang, L.Y.; Yu, K.L.; et al. Exploring volatile organic compounds in breath for high-accuracy prediction of lung cancer. *Cancers* **2021**, *13*, 1431. https://doi.org/10.3390/cancers13061431.
29. Choo, Y.J.; Lee, G.W.; Moon, J.S.; Chang, M.C. Application of non-contact sensors for health monitoring in hospitals: A narrative review. *Front. Med.* **2024**, *11*, 1421901. https://doi.org/10.3389/fmed.2024.1421901.
30. Nan, X.; Wang, X.; Kang, T.; Zhang, J.; Dong, L.; Dong, J.; Xia, P.; Wei, D. Review of Flexible Wearable Sensor Devices for Biomedical Application. *Micromachines* **2022**, *13*, 1395. https://doi.org/10.3390/mi13091395.
31. Moraes, F.M.D.; Silva, G.S. Noninvasive intracranial pressure monitoring methods: A critical review. *Arq. Neuro-Psiquiatr.* **2021**, *79*, 437–446. https://doi.org/10.1590/0004-282X-ANP-2020-0300.
32. Wolber, J.; Schuppert, A.; Mücke, M.; Sellin, J. Predicting disease outcomes from remote monitoring using machine learning: A systematic review. *BMC Med. Inform. Decis. Mak.* **2026**, *26*, 145. https://doi.org/10.1186/s12911-026-03495-0.
33. Skarysz, A.; Salman, D.; Eddleston, M.; Sykora, M.; Hunsicker, E.; Nailon, W.H.; Darnley, K.; McLaren, D.B.; Thomas, C.L.P.; Soltoggio, A. Fast and automated biomarker detection in breath samples with machine learning. *PLoS ONE* **2022**, *17*, e0265399. https://doi.org/10.1371/journal.pone.0265399.
34. Lee, A.R.; Lee, J.H.; Yoo, S.; Lee, H.Y.; Kim, H.H. Real-Time Estimation of Arterial Partial Pressure of Carbon Dioxide in Patients Undergoing General Anesthesia: Predictive Modeling Study. *JMIR Med. Inform.* **2025**, *13*, e64855. https://doi.org/10.2196/64855.
35. Woo, J.W.; Stapleton, O.; Luo, J.; Chuang, C.C.; Su, Y.; Sankararaman, S.; Mukherjee, E.; Sivakumar, N.; Calligy, K.; Duffy, S.; et al. AI-Driven SaO2 prediction from pulse oximetry and electronic health records. *BioData Min.* **2025**, *18*, 90. https://doi.org/10.1186/s13040-025-00511-3.
36. Cabanas, A.M.; Martín-Escudero, P.; Pagán, J.; Mery, D. Technical and regulatory challenges in artificial intelligence-based pulse oximetry: A proposed development pipeline. *Br. J. Anaesth.* **2025**, *134*, 1295–1299. https://doi.org/10.1016/j.bja.2025.02.014.
37. Ghazal, S.; Sauthier, M.; Brossier, D.; Bouachir, W.; Jouvet, P.A.; Noumeir, R. Using machine learning models to predict oxygen saturation following ventilator support adjustment in critically ill children: A single center pilot study. *PLoS ONE* **2019**, *14*, e0198921. https://doi.org/10.1371/journal.pone.0198921.

38. Hüser, M.; Lyu, X.; Faltys, M.; Pace, A.; Berger, D.; Hoche, M.; Hyland, S.L.; Yèche, H.; Burger, M.; Merz, T.M.; et al. RMS: A ML-based system for ICU respiratory monitoring and resource planning. *npj Digit. Med.* **2025**, *8*, 775. https://doi.org/10.1038/s41746-025-02081-4.
39. Si, T.; Song, Y.; Sun, P.; Xiao, J.; Zou, R.; Chen, Z. Non-contact SpO2 monitoring via multi-channel pulse signals from facial videos using machine learning. *Mach. Vis. Appl.* **2025**, *36*, 122. https://doi.org/10.1007/s00138-025-01736-2.
40. Urniezius, R.; Survyla, A.; Paulauskas, D.; Bumelis, V.A.; Galvanauskas, V. Generic estimator of biomass concentration for Escherichia coli and Saccharomyces cerevisiae fed-batch cultures based on cumulative oxygen consumption rate. *Microb. Cell Factories* **2019**, *18*, 190. https://doi.org/10.1186/s12934-019-1241-7.
41. Urniezius, R.; Survyla, A. Identification of Functional Bioprocess Model for Recombinant E. coli Cultivation Process. *Entropy* **2019**, *21*, 1221. https://doi.org/10.3390/e21121221.
42. Survyla, A.; Urniezius, R.; Vaitkus, V.; Levisauskas, D.; Jankauskaite, L.; Lukminaite, D.; Laucaityte, G. Noninvasive Continuous Tracking of Partial Pressure of Oxygen in Arterial Blood: Adapting Microorganisms Bioprocess Soft Sensor Technology for Holistic Analysis of Human Respiratory System. In Proceedings of the 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Karlsruhe, Germany, 23–25 September 2021; pp. 1–5. https://doi.org/10.1109/MFI52462.2021.9591182.
43. Kemesis, B.; Urniezius, R.; Kondratas, T.; Jankauskaite, L.; Masaitis, D.; Babilius, P. Bridging Functional Model of Arterial Oxygen with Information of Venous Blood Gas: Validating Bioprocess Soft Sensor on Human Respiration. In *Proceedings of the Intelligent and Safe Computer Systems in Control and Diagnostics*; Kowalczuk, Z., Ed.; Springer: Cham, Switzerland, 2023; pp. 42–51. https://doi.org/10.1007/978-3-031-16159-9_4.
44. Urniezius, R.; Galvanauskas, V.; Survyla, A.; Simutis, R.; Levisauskas, D. From Physics to Bioengineering: Microbial Cultivation Process Design and Feeding Rate Control Based on Relative Entropy Using Nuisance Time. *Entropy* **2018**, *20*, 779. https://doi.org/10.3390/e20100779.
45. Urniezius, R.; Kemesis, B.; Simutis, R. Bridging Offline Functional Model Carrying Aging-Specific Growth Rate Information and Recombinant Protein Expression: Entropic Extension of Akaike Information Criterion. *Entropy* **2021**, *23*, 1057. https://doi.org/10.3390/e23081057.
46. Urniezius, R.; Petrauskas, R.; Vaitkus, V.; Karimov, J.; Brazauskas, K.; Repsyte, J.; Kacerauskiene, E.; Harms, T.; Dargiene, J.; Ezerskis, D. Enhancing Prediction by Incorporating Entropy Loss in Volatility Forecasting. *Entropy* **2025**, *27*, 806. https://doi.org/10.3390/e27080806.
47. Kullback, S.; Leibler, R.A. On Information and Sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. https://doi.org/10.1214/aoms/1177729694.
48. Caticha, A.; Giffin, A. Updating Probabilities. In *Proceedings of the Bayesian Inference and Maximum Entropy Methods in Science and Engineering*; AIP Conference Proceedings; American Institute of Physics: College Park, MD, USA, 2006; Volume 872, pp. 31–42. https://doi.org/10.1063/1.2423258.
49. Giffin, A.; Urniezius, R. The Kalman Filter Revisited Using Maximum Relative Entropy. *Entropy* **2014**, *16*, 1047–1069. https://doi.org/10.3390/e16021047.
50. Giffin, A.; Urniezius, R. Simultaneous State and Parameter Estimation Using Maximum Relative Entropy with Nonhomogenous Differential Equation Constraints. *Entropy* **2014**, *16*, 4974–4991. https://doi.org/10.3390/e16094974.
51. Garcia-Ochoa, F.; Gomez, E.; Santos, V.E.; Merchuk, J.C. Oxygen uptake rate in microbial processes: An overview. *Biochem. Eng. J.* **2010**, *49*, 289–307. https://doi.org/10.1016/j.bej.2010.01.011.
52. Collins, J.A.; Rudenski, A.; Gibson, J.; Howard, L.; O'Driscoll, R. Relating oxygen partial pressure, saturation and content: The haemoglobin–oxygen dissociation curve. *Breathe* **2015**, *11*, 194–201. https://doi.org/10.1183/20734735.001415.
53. Nadler, S.B.; Hidalgo, J.U.; Bloch, T. Prediction of blood volume in normal human adults. *Surgery* **1962**, *51*, 224–232.
54. Ferrannini, E. The Theoretical Bases of Indirect Calorimetry: A Review. *Metabolism* **1988**, *37*, 287–301. https://doi.org/10.1016/0026-0495(88)90110-2.
55. Wick, K.D.; Matthay, M.A.; Ware, L.B. Pulse oximetry for the diagnosis and management of acute respiratory distress syndrome. *Lancet Respir. Med.* **2022**, *10*, 1086–1098. https://doi.org/10.1016/S2213-2600(22)00058-3.
56. Rubin, D.B. Inference and Missing Data. *Biometrika* **1976**, *63*, 581–592. https://doi.org/10.1093/biomet/63.3.581.
57. Burden, R.L.; Faires, J.D.; Burden, A.M. *Numerical Analysis*, 10th ed.; Cengage Learning: Boston, MA, USA, 2015. Available online: https://www.cengage.uk/c/numerical-analysis-10e-faires/9781305253667/ (accessed on 20 October 2026).
58. Spearman, C. The Proof and Measurement of Association between Two Things. *Am. J. Psychol.* **1904**, *15*, 72–101. https://doi.org/10.2307/1412159.
59. Fuller, W.A. *Measurement Error Models*; John Wiley & Sons: New York, NY, USA, 1987. https://doi.org/10.1002/9780470316665.
60. Efron, B.; Morris, C. Data Analysis Using Stein's Estimator and Its Generalizations. *J. Am. Stat. Assoc.* **1975**, *70*, 311–319. https://doi.org/10.1080/01621459.1975.10479864.

61. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. https://doi.org/10.1214/aos/1013203451.
62. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; pp. 785–794. https://doi.org/10.1145/2939672.2939785.
63. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. https://doi.org/10.1006/jcss.1997.1504.
64. Hanley, J.A.; McNeil, B.J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* **1982**, *143*, 29–36. https://doi.org/10.1148/radiology.143.1.7063747.
65. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-Generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA, 25 July 2019; pp. 2623–2631. https://doi.org/10.1145/3292500.3330701.
66. Varma, S.; Simon, R. Bias in error estimation when using cross-validation for model selection. *BMC Bioinform.* **2006**, *7*, 91. https://doi.org/10.1186/1471-2105-7-91.
67. Cawley, G.C.; Talbot, N.L.C. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *J. Mach. Learn. Res.* **2010**, *11*, 2079–2107.
68. Vincent, J.L.; Baron, J.F.; Reinhart, K.; Gattinoni, L.; Thijs, L.; Webb, A.; Meier-Hellmann, A.; Nollet, G.; Peres-Bota, D. Anemia and Blood Transfusion in Critically Ill Patients. *JAMA* **2002**, *288*, 1499–1507. https://doi.org/10.1001/jama.288.12.1499.
69. Bodley, T.; Chan, M.; Levi, O.; Clarfield, L.; Yip, D.; Smith, O.; Friedrich, J.O.; Hicks, L.K. Patient harm associated with serial phlebotomy and blood waste in the intensive care unit: A retrospective cohort study. *PLoS ONE* **2021**, *16*, e0243782. https://doi.org/10.1371/journal.pone.0243782.
70. Collins, G.S.; Moons, K.G.M.; Dhiman, P.; Riley, R.D.; Beam, A.L.; Van Calster, B.; Ghassemi, M.; Liu, X.; Reitsma, J.B.; van Smeden, M.; et al. TRIPOD+AI statement: Updated guidance for reporting clinical prediction models that use regression or machine learning methods. *BMJ* **2024**, *385*, e078378. https://doi.org/10.1136/bmj-2023-078378.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.