Federated AdaBoost for Survival Analysis

Preprint version

Federated AdaBoost for Survival Analysis

Oussama Harrak[1†], Bruno Casella[2†], Samuele Fonio[2†], Piero Fariselli[3], Gianluca Mittone[2], Cesare Rollo[3], Tiziana Sanavia[3], and Marco Aldinucci[2] *

[1] Polytech Clermont - Mathematical Engineering and Data Science Department, 2 Av. Blaise Pascal, 63100 Aubière - France harrak.oussama2002@gmail.com

[2] Alpha Research Group, Computer Science Department, University of Turin, C.so Svizzera 185, Turin - Italy {bruno.casella, samuele.fonio, gianluca.mittone, marco.aldinucci}@unito.it

[3] Computational Biomedicine Group, Medical Science Department, Via Santena 19, Turin - Italy {piero.fariselli, cesare.rollo, tiziana.sanavia}@unito.it

Abstract. This work proposes FedSurvBoost, a federated learning pipeline for survival analysis based on the AdaBoost.F algorithm, which iteratively aggregates the best local weak hypotheses. Our method extends AdaBoost.F by removing the dependence on the number of classes coefficient from the computation of the weights of the best model. This makes it suitable for regression tasks, such as survival analysis. We show the effectiveness of our approach by comparing it with state-of-the-art methods, specifically developed for survival analysis problems, on two common survival datasets. Our code is available at https://github.com/oussamaHarrak/FedSurvBoost.

Keywords: Federated Learning · Survival Analysis · Federated Survival Analysis · AdaBoost.

Survival Analysis (SA) is a multidisciplinary field encompassing statistical, mathematical, and machine-learning methodologies to analyze time-to-event data. Its primary focus is understanding the time until specific events occur, such as diseases, mechanical system failures, customer churn, or any other event of interest. SA has widespread application in many domains, including engineering, social sciences, and medicine. In healthcare, the event of major interest is death, but SA can deal with any other event affecting the patients, such as disease occurrence or recovery. However, due to the probably extended duration of observational studies, it is extremely common for patients to discontinue their participation in the study before the event of interest takes place. In such situations, this leads to the censoring phenomenon, i.e., the information information about the exact value of the event is incomplete.

* This work has been partly supported by the Spoke "FutureHPC & BigData" of the ICSC - Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union - NextGenerationEU, and by the Horizon2020 RIA EPI project (G.A. 826647).

† equal contribution.

SA builds on Machine Learning (ML) techniques to predict a survival function $S(t)$ representing the probability that an individual survives past a given point of time t. However, survival data often contains sensitive information, such as patients' clinical records and incomplete observations, known as censored data. Simultaneously, with increasing privacy constraints such as the European GDPR, there is a growing need for techniques to address these concerns and process inherently distributed data.

To address these privacy constraints, Federated Learning (FL) [1] has recently emerged as a prominent technique for preserving privacy while leveraging collaborative efforts among different parties (also referred to as *clients*). FL operates by exchanging locally trained models from clients rather than sharing the raw data. This collaborative approach facilitates the creation of a global model that typically outperforms the individual local models due to its enhanced generalization capabilities. This framework is particularly advantageous in the healthcare domain, where privacy concerns regarding inter-institutional sensitive data are prevalent.

In this work, we introduce FedSurvBoost, an algorithm that integrates SA into Federated Learning to address these challenges. Privacy preservation is achieved by respecting data locality, while collaboration is achieved by aggregating locally trained models into a shared ML model, retaining the whole federation's knowledge. FedSurvBoost exploits *AdaBoost.F* [2], a model-agnostic, federated ensemble learning algorithm allowing FL with classical, non-deep ML. Specifically, FedSurvBoost extends AdaBoost.F by modifying it to compute the best model's parameters for solving regression problems, effectively adapting it to SA tasks.

Our contributions can be summarized as follows:

- We propose FedSurvBoost, an extension of AdaBoost.F, for solving SA tasks.
- We show the effectiveness of FedSurvBoost through extensive experimentation on two survival benchmark datasets, proving that it outperforms state-of-the-art survival methods in terms of C-indexes and IBS scores.

In future works, we aim at extending the SA setting to the other proposed Boosting algorithms, such as PreWeak.F and DistBoost.F [2], to the SA setting.

1 Related Work

Federated Learning In its simplest configuration [1], each participant, referred to as a *client* within a federation, performs local training of a model. Afterwards, these participants share their trained models with a central server. This server then aggregates the received models with a specific strategy (e.g. averaging [1]) to form a global model and broadcasts it to the clients. This iterative procedure persists until the models converge. A typical FL scenario includes a server and many clients. Before the training begins, the server creates a Deep Learning (DL) model and broadcasts it to all the available clients. The clients then train their local model copy on the locally available data. The locally trained model's

parameters are then returned to the server, which aggregates them, according to some strategy (the most straightforward but still effective one is Federated Averaging [1]), into a single, shared DL model. One of the most straightforward, state-of-the-art aggregation strategies employed in current practice is Federated Averaging (FedAvg) [1], which calculates the average of all the locally trained model's parameters. Finally, the aggregated model is sent back to the clients, and the process continues iteratively until convergence. Due to its intrinsic DL-based nature, FL emerges in contexts requiring massive amounts of data, such as computer vision and natural language processing. However, some research works focused on enabling FL also for non-deep models [2]. This paper follows this trend: we propose FedSurvBoost, a modified version of *AdaBoost.F* suitable for SA tasks.

FedSurf: In the context of SA, distributed learning methodologies have recently gained attention, especially within the healthcare sector [3]. The existing approaches focus mainly on the adaptation of classical Cox models (either in their linear or deep version) to FL settings [4,5,6,7] or on the generation and sharing of synthetic survival data at the client level in a One-Shot learning fashion [8]. However, most of the proposed methods do not go beyond Cox's Proportional Hazard assumption, which may be too simplistic in large-scale distributed datasets. The state-of-the-art for federated SA is the Federated Survival Forests (FedSurF) [9] algorithm, which is an adaptation of the Random Survival Forests (RSF) to the federated scenario. RSF, one of the most successful ML models for survival analysis, recursively builds a set of binary survival trees to estimate the cumulative hazard function $H(t)$, which represents the total accumulated risk of an event occurring by time t. FedSurF is a federated ensemble learning approach in which the server aggregates the best-performing trees from the client-side RSF models. FedSurF executes three stages: local training, tree assignment, and tree sampling. During the local training, each client trains an RSF model on its local dataset. The trained model will encompass an ensemble of survival trees. During the tree assignment stage, the server chooses the number of trees that each client must transmit in order to achieve the desired number of trees. The server exploits a non-uniform probability proportional to the local dataset cardinality to promote greater inclusion of trees coming from clients with larger datasets. This mechanism resembles the FedAvg weighted contribution strategy. In the last stage of tree sampling, clients select the local trees to send to the server. The strategy for selecting the top-performing trees can be a uniform probability method or a metric-based method. Specifically, FedSurf-IBS is the adaptation of FedSurf using the inverse of the Integrated Brier Score (IBS) as a validation metric for the tree sampling step. Finally, the ensemble model is constructed by gathering the trees received from each client. Despite its effectiveness, FedSurf leverages only trees as weak learners. We overcome this restriction by adapting *Adaboost.F* to SA. Thanks to its model-agnostic nature, it enables the use of different weak learners, outperforming FedSurF.

2 Methodology

In this paper, we exploit the distributed implementation of the AdaBoost algorithm, namely *AdaBoost.F* [2] through an open-source implementation [10] and taking inspiration from adaptations of Adaboost to regression tasks. An open-source, computationally optimized implementation of AdaBoost.F is provided as an independent extension of the Intel OpenFL framework, namely OpenFL-extended [10]. Below, an explanation of AdaBoost.F in the classification task is provided to highlight the key points to be changed to adapt to the SA setting.

AdaBoost.F Let's suppose to have C clients. Each client starts with a local weak learner $h_c = \mathcal{A}(\mathbf{X_c}, \mathbf{y_c}, \mathbf{d_c})$ that must be trained on the local training set $(\mathbf{X_c}, \mathbf{y_c})$ and a set of weights $\mathbf{d_c}$ used to calculate the weighted error. For example, in a classification task, the weighted error of client c is calculated by:

$$\epsilon(h_c) = \sum_{i=1}^{n_c} d_c^i \mathbb{1}(y_c^i = h_c(x_c^i)), \tag{1}$$

where n_c is the number of examples of client c, and $\mathbb{1}$ is the indicator function. The final goal of Adaboost.F is to modify $\mathbf{d_c}$ using the information from other clients in a privacy-preserving process. At the beginning, it is initialized uniformly, $\mathbf{d} = (1, \ldots, 1) \in \mathbb{R}^{n_c}$.

The local dataset size is communicated to the server, and $h_c \leftarrow \mathcal{A}(\mathbf{X_c}, \mathbf{y_c}, \mathbf{d_c})$ is trained and sent to the server. The server collects all the weak learners in a vector of weak learners $\mathbf{h}$, which is sent to every client. Each client computes the error with all the models in $\mathbf{h}$, obtaining a vector of errors $\boldsymbol{\epsilon}$, and sends it to the server. For example, in a classification task, the error vector $\boldsymbol{\epsilon}$ would be:

$$\boldsymbol{\epsilon} = [\epsilon(h_c)]_{c=1}^{C} \tag{2}$$

The central server collects all the errors in a matrix $E^t \in \mathbb{R}^{C \times C}$. Using E^t, the server is able to detect which was the best-performing client at time t:

$$c^{t*} = \text{argmin}_c \sum_{c'=1}^{C} E_{cc'}^t, \tag{3}$$

and the overall error committed by the best model:

$$\epsilon^{t*} = \frac{1}{C} \sum_{c=1}^{C} E_{cc^{t*}}^t. \tag{4}$$

We leverage these quantities to calculate the following coefficients:

$$\alpha_t^* = log(\frac{1 - \epsilon^{t*}}{\epsilon^{t*}}) + log(K - 1), \tag{5}$$

where t is the round, K is the number of classes, and c^{t*} is the index of the best-performing client at the time t. These coefficients are used for two purposes: to update the vector $\mathbf{d_c}$ and to weight the final vote to predict.

In particular, α_t and c^{t*} are sent to each client. In this way, the clients can update the local weights for the weak learner:

$$\mathbf{d_c} \leftarrow [d_c^i \exp(-\alpha_t^*(-1)^{\mathbb{1}(h_{c^*}(x_i \neq y_i))})]_{i=1}^{n_c}. \tag{6}$$

These new weights will be used by the client's weak learner h_c to improve the local model performances. As a consequence, the process can restart until convergence.

At each iteration t, the server records the best model. In this way, the prediction of an example x can be performed using the best model at each iteration, with a majority vote policy. Specifically, the server predicts the class k with the highest number of occurrences in the best model predictions:

$$h_f(x) = \underset{k \in 1,\ldots,K}{\arg\max} \sum_{t=1}^{T} \alpha_t^* \mathbb{1}(h_{c^t *}(x) = k), \tag{7}$$

where T is the number of rounds. For more details about *Adaboost.F*, we invite the reader to refer to [2].

AdaBoost.F for SA Our effort has emerged in adjusting this algorithm to the SA setting, for which we need to redefine the error and its usage. In the specific, each weak learner h_c of client c predicts a survival time $\hat{t}_i$ for examples i (i.e. $h_c(x_i) = \hat{t}_i$) in their local dataset, and the error is calculated as:

$$\epsilon(h_c) = \sum_{i=1}^{n_c} d_c^i L_i, \tag{8}$$

$$L_i = \delta_i |\hat{t}_i - t_i| + (1 - \delta_i) \mathbb{1}_{\hat{t}_i - t_i < 0} |\hat{t}_i - t_i|. \tag{9}$$

where n_c is the number of examples for client c, $\hat{t}_i$ is the predicted survival time for the $i-$th sample and t_i is the true time predicted.

In SA, it is essential to distinguish between censored and uncensored data: for uncensored samples ($\delta_i = 1$), the error is the absolute difference between predicted and actual event times, reflecting direct observable discrepancies. Conversely, for censored samples ($\delta_i = 0$), we only consider a prediction to be in error if it underestimates the censored time. This is because predicting an event to occur before the last known observation contradicts the evidence we have, while predicting it to occur later does not [11].

At this point, the error vector that each client computed on each model received is the following:

$$\boldsymbol{\epsilon} = [\epsilon(h_c)]_{c=1}^{C} \tag{10}$$

and sent to the server. As before, we want to recompute $\mathbf{d_c}$ at the end of the aggregation. As in *Adaboost.F*, the server calculates c^{t*} and ϵ^{t*} as in (3) and (4) respectively, and we make sure to normalize ϵ^{t*} to be in between 0 and 1. However, inspired by [12], we replace α_t defined in (5) with the following:

$$\alpha_t = \frac{\epsilon^{t*}}{1 - \epsilon^{t*}}. \tag{11}$$

The smaller the α_t term, the more confidence we have in the best model c^{t*} and the better its predictions are. We can see that when the error ϵ^{t*} approaches the value of 0, α_t also approaches 0, indicating high confidence in the model's predictions c^{t*}. Conversely, when ϵ^{t*} approaches the value of 1, α_t approaches $+\infty$, indicating low confidence in the model predictions.

Then, α_t is sent to each client, and each client updates the distribution of the samples using the following formula, inspired by [12]:

$$\mathbf{d_c} = [d_c^i \alpha_t^{1-L_i^*}]_{i=1}^{n_c} \tag{12}$$

Where L_i^* is the error committed by the best model c^{t*} for sample i at round t.

For what concerns the prediction, it is clearly different from the classification task expressed in (7), as we are facing a regression problem. As a consequence, we substitute the prediction according to the original AdaBoost algorithm [12]:

$$h_f(x) = \sum_{t=1}^{T} \log(\frac{1}{\alpha_t}) h_{c^{t*}}(x). \tag{13}$$

The terms $\log(\frac{1}{\alpha_t})$ are used as weights for each best model c^{t*}. The weights assign higher positive values to models with lower errors (that are the closest to 0), indicating their importance in the ensemble. Conversely, models with higher errors (that are the closest to 1) receive large negative weights, indicating their low importance in the prediction :

$$\lim_{\epsilon^{t*} \to 0^+} \log(\frac{1}{\alpha_t}) = \lim_{\epsilon^{t*} \to 0^+} \log(\frac{1-\epsilon^{t*}}{\epsilon^{t*}}) = +\infty$$

and

$$\lim_{\epsilon^{t*} \to 1^-} \log(\frac{1}{\alpha_t}) = \lim_{\epsilon^{t*} \to 1^-} \log(\frac{1-\epsilon^{t*}}{\epsilon^{t*}}) = -\infty$$

3 Experiments

Our experiments aim to study the predictive performance of FedSurvBoost and compare it against FedSurF, serving as a baseline. We use the following experimental setting:

Testbed setup: FedSurvBoost experiments have been executed in a real distributed environment encompassing one server and C clients, each deployed on a dedicated server with an Intel®Xeon®processor (Skylake, IBRS, eight sockets of one core). The number of clients C ranges from one (standard centralized setting) to eight for scalability study purposes. We adopted OpenFL-extended [10] as the FL framework and Scikit-Learn to train the models. We ran the FedSurF competitor code with $C =\in \{1, 4, 8\}$ in a simulated federation deployed on a dedicated machine with the same hardware specifications listed previously.

Due to space constraints, we only report the average over three training runs for $C \in \{1, 4, 8\}$. Additional results, including a wider range of clients, the

averaged standard deviations, as well as the code required to replicate our experiments, are available at `https://github.com/oussamaHarrak/FedSurvBoost`.

Datasets and models: We focused on two benchmark survival datasets, i.e. the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) [13] comprising 1904 patients and 8 features, and the Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT) [4], comprising 9105 samples and 35 features. Data is split into training and testing data and distributed to each client, thus assuming independent and identically distributed (IID) data. We trained FedSurvBoost for 50 rounds with the following weak learners: CoxPH, Random Survival Trees (RST), and DeepSurv. CoxPH is a widely used semi-parametric model in SA.

Table 1. C-indexes of the models. Results (mean) are obtained with three averaged runs. Standard deviations are available in the repository.

Methods	Metabric Clients			Support Clients		
	Centralized	4	8	Centralized	4	8
FedSurF	0.622	0.637	0.629	0.824	0.834	0.829
FedSurF-IBS	0.622	0.639	0.627	0.824	0.826	0.826
FedAvg (PyCox)	0.630	0.643	0.628	0.844	0.841	0.844
RSF	0.630	0.645	0.642	0.791	0.810	0.780
FedSurvBoost weak learners						
CoxPH	0.646	0.653	0.656	0.831	0.835	0.838
RST	0.631	0.639	0.636	0.820	0.831	0.830
PyCox	**0.650**	**0.659**	**0.657**	**0.871**	**0.875**	**0.874**

Deepsurv is a non-linear version of the CoxPH model, for which we used the PyCox implementation (`https://github.com/havakv/pycox/`). RST is an adapted version of the traditional decision tree algorithm specifically designed for SA. The final model has been tested on a separate test dataset.

As evaluation metrics, we focus on the Concordance-index (c-index), which evaluates the capacity of the model to rank the individuals following their predicted risk of event, and the Integrated Brier Score (IBS), an extension of the mean squared error for censored data, that can measure discrimination and calibration. We tested our proposed approach with FedSurF and FedSurF-IBS, both aggregating RSFs, as well as with two naive approaches: one an approach based on RSF, and another one based on FedAvg and the PyCox model.

[4] Data obtained from http://hbiostat.org/data courtesy of the Vanderbilt University Department of Biostatistics.

Table 2. IBS scores of the models. Results (mean) are obtained with three averaged runs. Standard deviations are available in the repository.

Methods	Metabric Clients			Support Clients		
	Centralized	4	8	Centralized	4	8
FedSurF	0.175	0.171	0.171	0.179	0.180	0.179
FedSurF-IBS	0.175	0.170	0.170	0.179	0.156	0.157
FedAvg (PyCox)	0.168	0.168	0.169	0.127	0.128	0.129
RSF	0.164	0.167	0.171	0.138	0.140	0.139
FedSurvBoost weak learners						
CoxPH	**0.152**	**0.156**	**0.155**	0.140	0.142	0.140
RST	0.169	0.174	0.171	0.130	0.132	0.136
PyCox	0.165	0.170	0.169	**0.124**	**0.119**	**0.124**

Discussion: Tables 1 and 2 show the results of our experiments in terms of c-indexes and IBS scores. FedSurvBoost outperforms FedSurF (and its variant FedSurF-IBS) and the naive approach based on averaging the parameters of PyCox in both the metrics of c-index and IBS score (a lower value is better). Surprisingly, although AdaBoost.F was devised for aggregating non-gradient descent-based models, the best weak learner for FedSurvBoost is PyCox, in which a neural network parametrizes the predictor. This confirms the model-agnosticism of AdaBoost.F to produce a strong learner. Finally, FedSurvBoost shows a good scalability performance by outperforming the competitors on the METABRIC dataset when increasing the number of federation clients. Surprisingly, the algorithm does not suffer from sharding the dataset in more parties (sometimes c-indexes are even slightly better), probably due to the intrinsic uniform distribution of data of the datasets. The competitor's performance also confirms this insight.

4 Conclusion and Future Work

This paper proposes FedSurvBoost, a federated SA approach based on the adaptation of AdaBoost.F for regression tasks. Thanks to its model-agnosticism, we show the efficacy of FedSurvBoost by training different weak learners on two survival datasets. FedSurvBoost outperforms FedSurF, the state-of-the-art approach for SA, and the naive method based on FedAvg and a non-linear model such as PyCox. For future work, we aim to test FedSurvBoost on more survival datasets and to study its performance when dealing with non-IID censored data.

References

1. Brendan McMahan et al. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. PMLR, 2017.

2. Mirko Polato et al. Boosting the federation: Cross-silo federated learning without gradient descent. In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pages 1–10. IEEE, 2022.
3. Francesco Cremonesi et al. The need for multimodal health data modeling: A practical approach for a federated-learning healthcare platform. *Journal of Biomedical Informatics*, 141:104338, 2023.
4. Mathieu Andreux et al. Federated survival analysis with discrete-time cox models. *arXiv preprint arXiv:2006.08997*, 2020.
5. Akira Imakura et al. Dc-cox: Data collaboration cox proportional hazards model for privacy-preserving survival analysis on multiple parties. *Journal of Biomedical Informatics*, 137:104264, 2023.
6. Xuan Wang, Harrison G Zhang, Xin Xiong, Chuan Hong, Griffin M Weber, Gabriel A Brat, Clara-Lea Bonzel, Yuan Luo, Rui Duan, Nathan P Palmer, et al. Survmaximin: robust federated approach to transporting survival risk prediction models. *Journal of biomedical informatics*, 134:104176, 2022.
7. Carlotta Masciocchi, Benedetta Gottardelli, Mariachiara Savino, Luca Boldrini, Antonella Martino, Ciro Mazzarella, Mariangela Massaccesi, Vincenzo Valentini, and Andrea Damiani. Federated cox proportional hazards model with multicentric privacy-preserving lasso feature selection for survival analysis from the perspective of personalized medicine. In *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 25–31. IEEE, 2022.
8. Cesare Rollo et al. Syndsurv: A simple framework for survival analysis with data distributed across multiple institutions. *Computers in Biology and Medicine*, 172:108288, 2024.
9. Alberto Archetti and Matteo Matteucci. Federated survival forests. In *International Joint Conference on Neural Networks, IJCNN 2023, Gold Coast, Australia, June 18-23, 2023*, pages 1–9. IEEE, 2023.
10. Gianluca Mittone et al. Model-agnostic federated learning. In *Euro-Par 2023: Parallel Processing - 29th International Conference on Parallel and Distributed Computing, Limassol, Cyprus, August 28 - September 1, 2023, Proceedings*, September 2023.
11. Torsten Hothorn, Peter Bühlmann, Sandrine Dudoit, Annette Molinaro, and Mark J Van Der Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 2006.
12. Harris Drucker. Improving regressors using boosting techniques. In *Icml*, volume 97, page e115. Citeseer, 1997.
13. Jared L. Katzman et al. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):24, 2018.