AutoFinSurv: An Automatic XGBoost-Cox Optimization Framework for Financial Survival Analysis

Rong-Lin Jian
Institute of Intelligent Systems, National Yang Ming Chiao Tung University
Hsinchu, Taiwan
ron0410530@gmail.com

Abstract

We present **AutoFinSurv**, a reproducible GPU-accelerated framework for financial survival analysis based on the XGBoost-Cox model. Our method integrates temporal data splitting, unified preprocessing, and Optuna-based hyperparameter optimization. Experiments on the official FinSurvival Challenge datasets demonstrate that AutoFinSurv consistently improves concordance index (C-index) across multiple event pairs, achieving an overall mean C-index of 0.8483 in the Codabench Development Phase and 0.8490 on the Final Phase leaderboard, ranking among the top submissions.

ACM Reference Format:
Rong-Lin Jian. 2025. AutoFinSurv: An Automatic XGBoost-Cox Optimization Framework for Financial Survival Analysis. In . ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The FinSurvival Challenge [5] benchmarks predictive models for time-to-event outcomes in decentralized finance. Each instance involves a financial "index event" (e.g., Borrow, Deposit, Withdraw) and predicts the probability of subsequent "outcome events" (e.g., Repay, Liquidated). Unlike static classification, survival modeling must capture temporal dependencies while handling censored samples [3]. Our goal is to build a transparent, modular system that can be reproduced end-to-end on GPU hardware.

2 Methodology

2.1 Temporal Split and Preprocessing

For each event pair, data before July 1, 2023−30 days were used for training, and data after July 1, 2023+30 days for validation. Categorical fields were reduced to their ten most frequent categories (+"Other") and one-hot encoded, while numerical features were standardized using `StandardScaler`. This ensures alignment of train/test features without leakage.

Training and Validation Split. To maintain balanced event−censor distributions, all train/validation splits were stratified by event status (`status`) to preserve consistent positive−negative ratios. A fixed random seed (`seed=42`) was used for all experiments to guarantee reproducibility.

2.2 Model

We employed the XGBoost framework [2] with the Cox proportional-hazards objective [3]. Each label was converted to signed time:

$$y_i = \begin{cases} +t_i, & \text{if the event occurred,} \\ -t_i, & \text{if censored.} \end{cases} \tag{1}$$

The Cox partial likelihood loss was defined as:

$$\mathcal{L}_{Cox} = -\sum_{i:\delta_i=1} \left(\eta_i - \log \sum_{j \in R_i} e^{\eta_j} \right), \tag{2}$$

where $\eta_i = f_\theta(x_i)$ is the predicted log-risk score. Training employed the GPU-accelerated tree builder (`tree_method=gpu_hist`) on an NVIDIA Titan RTX.

2.3 Hyperparameter Optimization

Baseline Configuration. Before optimization, a fixed baseline configuration was used across all event pairs for fair comparison: learning rate = 0.05, maximum depth = 6, subsample = 0.80, colsample_bytree = 0.8, min_child_weight = 5, reg_lambda = 1.0, and reg_alpha = 0.1. This served as the reference for subsequent Optuna searches.

Optuna [1] performed Bayesian optimization over the following space: learning_rate [0.03–0.05], max_depth [4–6], subsample [0.75–0.9], colsample_bytree [0.7–0.9], min_child_weight [3–8], reg_lambda [0.5–2.0], reg_alpha [0.05–0.5]. Each study ran 20 trials × 1000 boosting rounds with early stopping (50 rounds).

Experiment Logging and Artifacts. During tuning, each task's results were logged in real time: C-index scores and parameter configurations were appended to a log file, summarized in CSV format, and stored as JSON for reproducibility. Additionally, a bar plot (`xgb_refined_tuning_delta.png`) was automatically generated to visualize per-task ΔC-index improvements.

3 Experiments

3.1 Hardware and Setup

- GPU: NVIDIA Titan RTX (24 GB VRAM)
- OS: Ubuntu 22.04 + CUDA 12.1
- Libraries: XGBoost 2.1.1 [6], Optuna 3.6 [1], Scikit-learn 1.5, Lifelines [4]
- Dataset: Official FinSurvival `participant_data-001`
- Metric: Concordance index (C-index)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Comparison of AutoFinSurv performance on the Codabench Development and Final Phases. Scores were computed by the official Codabench scoring program using a deterministic 100,000-record subsample per dataset.

Dataset	Baseline (Dev)	AutoFinSurv (Dev)	AutoFinSurv (Final)
Repay→Deposit	0.8000	0.8092	0.8105
Repay→Withdraw	0.7874	0.7850	0.7863
Repay→Liquidated	0.7036	0.7138	0.7162
Repay→Borrow	0.8076	0.8131	0.8147
Deposit→Repay	0.8948	0.8979	0.8973
Deposit→Withdraw	0.8495	0.8461	0.8466
Deposit→Liquidated	0.9065	0.9116	0.9104
Deposit→Borrow	0.8924	0.8943	0.8962
Withdraw→Repay	0.9415	0.9434	0.9407
Withdraw→Deposit	0.8388	0.8372	0.8371
Withdraw→Liquidated	0.9671	0.9704	0.9672
Withdraw→Borrow	0.9354	0.9510	0.9488
Borrow→Repay	0.8314	0.8288	0.8295
Borrow→Deposit	0.7975	0.7998	0.8020
Borrow→Withdraw	0.7566	0.7887	0.7884
Borrow→Liquidated	0.7466	0.7840	0.7919
Mean	**0.8410**	**0.8483**	**0.8490**

Each task required approximately 1.2 GPU-hours, totaling around 320 GPU-hours for the full set of event pairs.

3.2 Leaderboard Performance (Codabench Evaluation)

Both Development (public) and Final (private) Phases on Codabench use the same evaluation pipeline, which deterministically subsamples 100,000 test records per dataset for computing the C-index. Table 1 compares results across Baseline, AutoFinSurv (Development), and AutoFinSurv (Final) phases. The mean C-index improved from 0.8410 (baseline) to 0.8483 (Development) and remained stable at 0.8490 on the Final Phase leaderboard, demonstrating strong generalization between public and private sets. The tuned parameters for each event pair were reused via `xgb_cox_submit.py` to produce final predictions on the official test features.

4 Discussion and Conclusion

AutoFinSurv achieved consistent C-index gains across most event pairs, with the largest improvements on Borrow→Withdraw and Borrow→Liquidated. This confirms that Optuna-driven tuning effectively enhances model robustness across heterogeneous financial domains. The framework's unified preprocessing, temporal segmentation, and GPU-accelerated optimization enable reliable generalization for survival prediction. Its modular design supports future integration with FPBoost or DeepCoxCC ensembles. All results are fully reproducible through logged CSV, JSON, and TXT artifacts.

Acknowledgment

We thank the FinSurvival Challenge organizers and the Codabench platform for providing resources and baseline codes that made this work possible.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), 2623–2631. doi:10.1145/3292500.3330701

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), 785–794. doi:10.1145/2939672.2939785

[3] David R. Cox. 1972. Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)* 34, 2 (1972), 187–220.

[4] Cameron Davidson-Pilon. 2023. Lifelines: Survival Analysis in Python. https://lifelines.readthedocs.io/.

[5] FinSurvival Organizers. 2024. FinSurvival Challenge 2024: Benchmarking Time-to-Event Prediction in Decentralized Finance. In *Codabench Competition Report*. https://www.codabench.org/competitions/10561/

[6] XGBoost Developers. 2024. XGBoost Documentation v2.1.1. https://xgboost.readthedocs.io/en/stable/.