Fairness in Survival: Equitable Survival Modelling for Allogeneic HCT

Jakob Jerše (#s1156064)
Dariga Shokayeva (#s1158777)
Daniel Duhnev (#s1158773)

1 Introduction

1.1 Background Context

Allogeneic Hematopoietic Cell Transplantation (HCT)[11] is a critical medical procedure where a patient receives healthy hematopoietic stem cells from a donor to replace their own dysfunctional stem cells in order to rebuild their immune system. Current predictive models often fail to account for disparities linked to socioeconomic status, race, and geography, leading to inequitable outcomes.
The Kaggle competition *"CIBMTR - Equity in post-HCT Survival Predictions"*[5] seeks to enhance the prediction of survival rates for patients undergoing Allogeneic HCT and to develop models that are both accurate and fair across diverse racial groups. The challenge used synthetic data generated with SurvivalGAN[20][21] that mirrors real-world data. The high level goal of the challenge is to improve patient care, optimize resource allocation, and foster trust in healthcare by ensuring equitable survival predictions.

1.2 Understanding the evaluation metric

Models were evaluated using the Stratified Concordance Index (C-index) metric. The C-index is defined as:

$$C\text{-index} = \frac{\sum_{i,j} I_{T_j<T_i} \cdot I_{n_j>n_i} \cdot \delta_j}{\sum_{i,j} I_{T_j<T_i} \cdot \delta_j}$$

where: n_i is the risk score of a unit i; $I_{T_j<T_i} = 1$ if $T_j < T_i$ else 0; $I_{n_j>n_i} = 1$ if $n_j > n_i$ else 0; and $\delta_j = 1$ if patient j had the event, else 0. This metric measures, for pairs of patients where it is known one had the event sooner, whether the model correctly attributes a higher risk score to the patient with the earlier event[8]. A value of 1 means risk scores perfectly match the true order of events, while a value of 0.5 means the scores are as good as random. The stratification adjusts the C-index by calculating it separately for each racial group, taking the mean minus the standard deviation across groups to ensure equitable performance; a high mean with a low standard deviation yields a better score, emphasizing fairness across diverse demographics.

2 Method

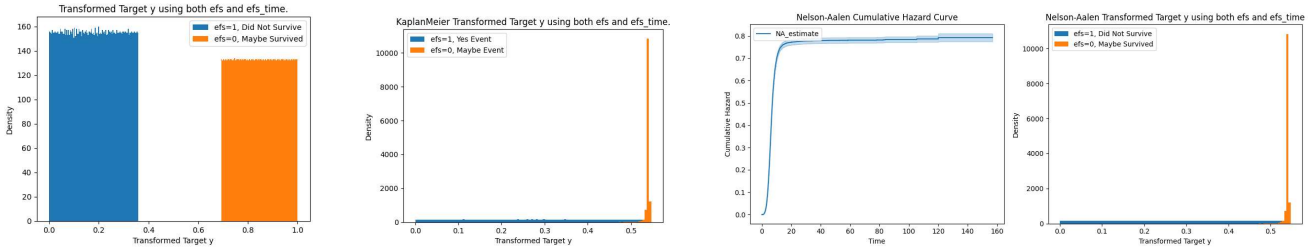
2.1 Exploratory Data Analysis

Firstly, Exploratory Data Analysis (EDA) was conducted to familiarize ourselves with the dataset. To begin with, we examined the data and found there were 57 features in total - 35 categorical and 22 numerical. From this analysis it was noted that some features are highly correlated with a correlation close to 1. Furthermore, it was noted that most numerical features, except for *donor_age* and *age_at_hct* did not have continuous distributions. Instead, they had only a handful of possible values ranging from only 2 values for features like *hla_match_c_high*, up to 12 values for features like *year_hct*. These observations were used for later Data Preprocessing and Feature Engineering (FE). Moreover, we observed an anomaly in the patient age distribution, specifically noting 1005 entries with an age of 0.0044 years. This peculiarity was also discussed in a Kaggle post[1]. Later, we aimed to address this anomaly by evenly distributing these data points across cross-validation folds to avoid bias[].

2.2 Data Preprocessing

We examined the data and identified a number of areas to consider for data preprocessing. First, we noticed that a number of categorical disease features such as *diabetes* and *arrhythmia* had both empty and "Not done" values. To improve the data quality for these features, we substituted all empty values for "Not done". We also had to decide how to deal with NAN values for numerical features. We noted that Gradient Boosting Decision Trees (GBDT) handle these values natively.

Most importantly, we had to decide how to combine the two target variables efs and efs_time. We noted that this ML problem is different to classical supervised problems, where instead of predicting a target label the model had to assign a risk score per patient representing the likelihood of an event such as relapse or death happening. The problem is made harder with censoring[19], meaning incomplete information is available about the survival time of some individuals. The C-index is then used to evaluate how well the scores rank the survival outcomes of patients.
We identified three approaches to transform efs and efs_time - combining the two with a simple manual transformation, using the Kaplan-Meier estimator[10], and using the Nelson-Aalen estimator[3].
The first and simplest approach included manually combining the two variables into a single target. This was done by shifting censored cases ($efs = 0$) by adding the difference between the max event time ($efs = 1$) and min censored time, ranking all values, further shifting censored ranks by half the dataset length, and normalizing between 0 and 1. The result can be seen in Figure 1a



(a) Baseline Density Distribution of *efs*

(b) Post-Kaplan-Meier Density Distribution

(c) Nelson-Aalen Cumulative Hazard Curve & Density Distribution

Figure 1: Comparison of Survival Target Transformations Across Kaplan-Meier and Nelson-Aalen Methods.

In order to use the Kaplan-Meier and Nelson-Aalen estimators, we used the $lifelines$ library[7] for survival analysis in Python. On the one hand, The Kaplan-Meier estimator computes the survival function, giving a stepwise probability of surviving past a given time, which we transform into a risk score for GBDT models to predict survival outcomes. The resulting continuous target can be seen in Figure 1b. On the other hand, the Nelson-Aalen estimator computes the cumulative hazard, which was converted into a continuous event probability resulting in a similar distribution seen in Figure 1c. Both transformations enhance survival predictions by approximating event risks.

2.3 Model Selection and Training

Our baseline notebook[9] included an ensemble of XGBoost and CatBoost models. We chose this notebook because it involved no feature engineering and only basic data preprocessing, making it a solid starting point. During our research[22], we found that GBDT models typically perform well with tabular data [4] and often outperform Neural Networks (NNs). They offer a more interpretable solution (providing clearer insights into feature importance and decision paths) and can achieve similar or better results than NNs.
Additionally, we discovered that GBDT models are a popular choice in Kaggle competitions[2] because they handle missing values natively, can work with both categorical and numerical features, train quickly (compared to deep learning models), are highly effective, and offer more interpretability compared to NNs.
We first trained the baseline setup as it seemed simple enough to provide a good starting point without requiring complex code. We opted for this approach, as it was the fastest way to give us a solid baseline, rather than training a linear regression model or a single GBDT model.

2.4 Model Refinement

Initially, we placed a lot of focus into feature engineering[12]. The features and their impact on the score are summarised later in Table 1 and 2. We proceeded with tuning the model using RandomizedSearchCV to find the best parameter subset for each model. RandomizedSearchCV was chosen over GridSearchCV due to being faster and providing a good balance between efficiency and performance. Additionally, testing different fold sizes, we ended using 10 folds which proved to have the highest CV score.
Next, we added early stopping for each model. During cross-validation, we monitored performance on the validation set and we observed from the train logs that in each fold, the model was overfitting to the training data — at some point validation error started increasing while training error kept decreasing. To address this, we implemented early stopping, ensuring that training stopped before overfitting occurred, allowing predictions to be made using the best-performing model.
LightGBM was also added to the ensemble[16] due to its efficiency handling large datasets and native support for

categorical features[2]. Finally, we experimented with model stacking instead of a simple ensemble. The key idea is that a meta-model learns to optimally combine the strengths of the base models, leveraging their complementary insights to improve overall performance. We trained a meta-model on the predictions of XGBoost, CatBoost, and LightGBM by first using simple Ridge Regression, where the alpha value was determined via GridSearchCV, and then tested XGBoost as a meta-model, tuning its parameters with RandomizedSearchCV.

3 Results

3.1 Experiments in Feature Engineering

The FE-related changes that were kept in the final notebook are summarized along with their impact on CV performance in Table 1. Many experiments related to FE that we tried failed to show any improvements or worsened overall CV performance. We present a summary of some of them in Table 2. In Table 1 and Table 2, the sign ↑ means "improvement" and the sign ↓ means "decrease" in ensemble performance based on CV score.

Table 1: Successful FE Changes in The Final Notebook

Feature/Change	**Impact on CV Score**
Removed HLA-related features excluding 10-resolution	< 0.001 ↑
Grouped *conditioning_intensity* into "Reduced Intensity", "MAC", and "Unknown"	≈ 0.001 ↑
Added *age_difference* (donor's age - recipient's age)	≈ 0.001 ↑
Added *sex_match_binary* (1 for same-sex, 0 otherwise)	≈ 0.001 ↑
Added *num_diseases* (count of diseases marked "Yes")	< 0.001 ↑
Imputed missing values in disease-related columns with "Not done"	< 0.001 ↑
Added *multiple_diseases* (binary: *num_diseases* > 1)	< 0.001 ↑

Table 2: FE Unsuccessful Experiments

Feature/Change	**Impact on CV Score**
Adding missing value indicators	↓
Imputation of NaN values with median/mode	↓
Aggregate features (*patient_age* * *comorbidity_score*, *age_ratio*)	Same/ minor ↓
Dropping less important HLA features (e.g., *hla_match_dqb1_low*)	↓
PCA on HLA-match features	↓
Adding a weighted score for high/low-HLA-match features	↓
Keeping high-resolution HLA features only	↓
Flagging years during COVID	↓

3.2 Target Transformations & Hyperparameter Tuning

Both estimators of *efs* and *efs_time* into a single target y boosted the performance of our model compared to the baseline transformed target. After FE, we also transformed numerical features to categorical for CatBoost and used RandomizedSearchCV to tune the hyperparameters of each model and use the optimal ones for training. Comparison of performances in terms of Cross Validation (CV) and LeaderBoard (LB) scores for these transformations after hyperparameter tuning along with the post-FE baseline is presented in Table 3.

Table 3: Scores' Comparison of Target Transformation Methods, Hyperparameter Tuning, and Early Stopping

Method	**CV**	**Public LB**	**Private LB**
Initial Baseline	0.66804	0.66964	0.68122
Post FE-Baseline	0.66957	0.67167	0.68107
Kaplan-Meier + RandomizedSearchCV	0.67531	0.68326	0.68311
Nelson-Aalen + RandomizedSearchCV	0.67568	0.68308	0.68326
Nelson-Aalen + RandomizedSearchCV + Early Stopping (1st run)	0.67596	0.68278	0.68340
Nelson-Aalen + RandomizedSearchCV + Early Stopping (2nd run)	0.67594	0.68302	0.68354
Previous Implementation + LightGBM * [16]	0.67614	N/A	N/A
Stacking with Ridge Regression as a Meta-Learner * [14]	0.67556	N/A	N/A
Stacking with XGBoost as a Meta-Learner * [15]	0.67547	N/A	N/A

* marks the submission made after the Kaggle deadline; LB scores are not available

3.3 Early Stopping

Early stopping achieved prediction improvements in CV scores for each model on its own and also on the model emsemble. Performance improvement for the model ensemble due to early stopping is also shown in Table 3.Firstly, early stopping rounds with different settings were tested for XGBoost and CatBoost. In the 2nd run, both were set to a local optimal of 75 rounds.

3.4 Changes to Model Ensemble

Adding LightGBM to an ensemble led to an increase of CV score as shown in 3. We also tried stacking models [14] [15] (XGBoost, CatBoost, and LightGBM) instead of ensembling that led to CV score decrease shown in Table 3.

3.5 Overall Performance

Figure 2 shows the changes in scores and the best scores achieved during the competition across different metrics: CV score, Public LB score, and Private LB score. The submission #6 is not included due to an exception error.

Submission ID	CV	Public LB	Private LB
7	0.67636	0.68228	0.68342
9	0.67531	0.68326	0.68311

Table 4: Kaggle Ranked Submissions in the Competition

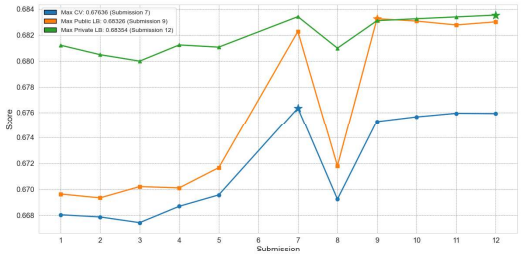


Figure 2: Submission Results with Scores

Table 4 shows the submissions that were used for the leaderboard. Submission #7 was pre-selected by us and submission #9 was automatically chosen by Kaggle for its highest public LB score. Submission #12 had the overall highest performance, with a private LB score of 0.68354.

4 Discussion

Replacing the baseline target variable transformations with Kaplan-Meier or Nelson-Aalen estimators showed the largest improvement in CV and LB scores (≈ 0.012), as presented in Table 3 and Figure 2. These estimators showed to be effective for censored data, providing unbiased survival probability estimates. Minor improvements were also observed from early stopping, feature engineering (e.g., introducing *age_difference*, *conditioning_intensity_grouped*), and adding LightGBM to the CatBoost and XGBoost ensemble. This confirms our assumptions that early stopping helps to prevent overfitting, appropriate feature engineering can help to better capture key patterns in the data better, and that LightGBM is well-suited for our ensemble approach.

We tried a number of approaches that did not improve our score. Firstly, as seen in Table 2, several approaches in FE failed to improve the model scores. Addressing an anomaly in the age distribution of patients in Submission 8, where specific data points were evenly distributed across folds, resulted in a decline in prediction performance[13]. To explain the sudden drop in Figure 2, both submissions #7 and #8 were built in parallel from submission #5. Attempts at model stacking using XGBoost, CatBoost, and LightGBM ensemble and either Ridge Regression or XGBoost as a meta-learner also resulted in score drops compared to simple ensembles. Some changes were implemented after the Kaggle competition deadline, meaning no leaderboard scores were available for these submissions. This is further discussed in Section 7.2.

5 Conclusion

The initial baseline model, an ensemble of XGBoost and CatBoost, achieved a CV score of 0.66804, a Public LB score of 0.66964, and a Private LB score of 0.68122. To improve the model, we transformed target variables using the Nelson-Aalen estimator for censored survival data, conducted feature engineering, implemented early stopping to prevent overfitting, and expanded the ensemble with LightGBM for improved predictive power.
These improvements led to our best submission, with a CV score of 0.67594, a Public LB score of 0.68302, and a Private LB score of 0.68354, reflecting gains of approximately 0.013 on the Public LB and 0.002 on the Private LB compared to the baseline.
Our approach centred on an ensemble of XGBoost, CatBoost, and LightGBM. We hypothesize that incorporating a Cox Proportional Hazards model[6] and a deep learning model like DeepSurv[17][18] could further boost accuracy.

References

[1] AmbrosM. The weirdness of the patient age distribution, December 2024. Kaggle discussion post.

[2] APXML. Main models to know for tabular data on kaggle, 2024.

[3] Ørnulf Borgan. Nelson–Aalen Estimator. Retrieved from McGill University Epidemiology website.

[4] Harald Carlens. State of competitive machine learning in 2022. *ML Contests Research*, 2023. https://mlcontests.com/state-of-competitive-machine-learning-2022.

[5] CIBMTR. CIBMTR - Equity in post-HCT Survival Predictions. `https://www.kaggle.com/competitions/equity-post-HCT-survival-predictions/overview`, 2024. Kaggle Competition.

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

[7] Cameron Davidson-Pilon et al. lifelines: Survival Analysis in Python. `https://pypi.org/project/lifelines/`, 2024. Python package for survival analysis, including Kaplan-Meier, Nelson-Aalen, and regression methods, accessed on March 13, 2025.

[8] Chris Deotte. How to get started - understanding the metric. `https://www.kaggle.com/competitions/equity-post-HCT-survival-predictions/discussion/550003`, December 2024. Kaggle Discussion Post.

[9] Chris Deotte. XGBoost CatBoost Baseline - [CV 668 LB 668]. `https://www.kaggle.com/code/cdeotte/xgboost-catboost-baseline-cv-668-lb-668`, December 2024. Kaggle Code Notebook.

[10] Manish Kumar Goel, Pardeep Khanna, and Jugal Kishore. Understanding survival analysis: Kaplan-Meier estimate. *International Journal of Ayurveda Research*, 1(4):274–278, October 2010.

[11] Boglarka Gyurkocza, Andrew Rezvani, and Rainer F. Storb. Allogeneic hematopoietic cell transplantation: the state of the art. *Expert Review of Hematology*, 3(3):285–299, June 2010.

[12] Daniel Duhnev Jakob Jerse, Dariga Shokayeva. Feature engineering experiments. `https://www.kaggle.com/code/darigashokayeva/submission-n5-mlip-33-quack-overflow?scriptVersionId=227619278`, 2024. Accessed: 2025-03-14.

[13] Daniel Duhnev Jakob Jerse, Dariga Shokayeva. Equalize special age indices distribution for each fold. `https://www.kaggle.com/code/daniduhnev/copy-xgboost-catboost-baseline-mlip-33?scriptVersionId=224460318`, 2025. Accessed: 2025-03-14.

[14] Daniel Duhnev Jakob Jerse, Dariga Shokayeva. Stacking with ridgeregression as meta learner. `https://www.kaggle.com/code/daniduhnev/xgboost-catboost-lightgbm-mlip-33-stacking1`, 2025. Accessed: 2025-03-14.

[15] Daniel Duhnev Jakob Jerse, Dariga Shokayeva. Stacking with xgboost as meta learner. `https://www.kaggle.com/code/daniduhnev/xgboost-catboost-lightgbm-mlip-33-stacking2`, 2025. Accessed: 2025-03-14.

[16] Jakob Jerse, Dariga Shokayeva, and Daniel Duhnev. Final submission with final model architecture. `https://www.kaggle.com/code/daniduhnev/project-1-mlip-33-final`, 2024. Accessed: 2025-03-14.

[17] Jared L. Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. DeepSurv: Personalized Treatment Recommender System Using A Cox Proportional Hazards Deep Neural Network. *BMC Medical Research Methodology*, 18(1):24, 2018.

[18] Katzman, Jared Lee. DeepSurv: Deep learning approach to survival analysis. `https://github.com/jaredleekatzman/DeepSurv`, 2025. GitHub repository, accessed March 14, 2025.

[19] K. M. Leung, R. M. Elashoff, and A. A. Afifi. Censoring issues in survival analysis. *Annual Review of Public Health*, 18:83–104, 1997.

[20] Alexander Norcliffe, Bogdan Cebere, Fergus Imrie, Pietro Lio, and Mihaela van der Schaar. Survivalgan: Generating time-to-event data for survival analysis, 2023.

[21] Alexander Norcliffe and van der Schaar Lab. SurvivalGAN: Generating Time-to-Event Data for Survival Analysis. `https://github.com/vanderschaarlab/survivalgan`, 2023.

[22] Kriti Singh. Why gradient boosting tree-based algorithms have become a popular choice for tabular data, 2025.