Evaluating Tree-based Models for Survival Analysis

Caijun Qin

Statistics, University of Florida

Abstract

Can classical machine learning algorithms accurately model survival analysis models? Survival analysis aims to study the rate of an event occurring to a sample or population over a duration of time points. Many applications range from clinical trials to predict terminal illness to the deprecation of factory machines to decide when to replace equipment. Survival analysis actually includes classification problems within the study. This includes predicting whether an event may occur any some time point and estimating how much time is left before an event occurs with some margin of error. In machine learning, classification algorithms utilizing decision trees have been proven to be robust and effective models. This study seeks to compare tree-based machine learning models to traditionally used models in survival analysis.

Keywords: survival analysis, machine learning, decision tree, boosting

Introduction

Survival analysis is a branch of statistics analyzing the failure or mortality rate of a population over some interval of time points. Oftentimes, this field has seen applications in estimating death rate for patients of terminal illnesses, machine failure, time to default for high-risk debtors, and more. Different estimator models have been instrumental in this field. Particularly, this study focuses on two foundational models, namely Kaplan-Meier and Cox Proportional-Hazards. The former will be more focused on as a representative of traditionally used models for survival analysis.

The *Kaplan-Meier (KM)* model, synomously the *product limit* model, estimates the fraction of subjects remaining at each point in time for a duration, symbolized as $\{t_0, t_1, t_2, \dots, t_N\}$. At t_0, it is assumed that all subjects are initially survivors, meaning no events have occurred yet. This yields $\hat{S}(t_0) = 1$. At some arbitrary time *t*, the fraction of remaining subjects can be modeled by a *survival function* $\hat{S}(t) = \Pi_{i:t_i \le t}\left(1 - \frac{d_i}{n_i}\right)$.

Here, $\hat{S}(t)$ denotes the fraction of all initial subjects that endured the event at time *t* or earlier. The variables d_i and n_i denote the number of subjects that endured the event only at time t_i and the current number of surviving subjects just up to t_i. As implied with hat in notation, $\hat{S}(t)$ is an

estimator, as the initial group of subjects is assumed to be a sample of some larger population, as is often the case in most survival analysis studies actually experimented. This model is recursive in nature, meaning the probability of failure can be computed from the survival rate from the previous time point and the proportion of failures from the current surviving subjects. The model can be re-expressed as the following.

$$\hat{S}(t_i) = \left[\Pi_{k=0}^{i-1}\left(1 - \frac{d_k}{n_k}\right)\right]\left(1 - \frac{d_i}{n_i}\right) = S(t_{i-1})\left(1 - \frac{d_i}{n_i}\right)$$

The KM model has several has several benefits to comprehensibility and ease of use. The simplicity of the survival function makes KM very explanatory. The survival rate any arbitrary time *t* on the cumulative history of survival rates from past time points. Furthermore, KM is nonparametric and therefore can be flexibly used when the distribution of events, if there is a underlying distribution at all, is known. The downside of KM is the inability to consider additional factors as part of the prediction process.

Dataset Summary

The dataset originates as a collected and cleaned version from a case study paper *Project Success Prediction in Crowdfunding Environments* (Li et al., 2017). From now, let the dataset in its original form be denoted as *D*. There are 4175 rows in *D*, each representing a specific project under Kickstarter on a specific day since being funded. *D* has 56 columns, two (2) of which being time (*day_succ*) and indicator variable for event (*Status*). day_succ has type integer, forms the inclusive range $[1,60]$, and denotes how many days have passed since crowdsourcing began. *Status* is a binary indicator variable, taking on either 1 or 2 to denote whether a specific project was ongoing or failed on a given day. All other 2-level factors take values of 0 or 1.

As a basis for comparison, the KM model with the simplest possible configuration is fitted. This model uses time (in days) as the only factor and applies no additional transformations to the estimator model. The base model, with relative risk over time, is plotted below in Figure 1.

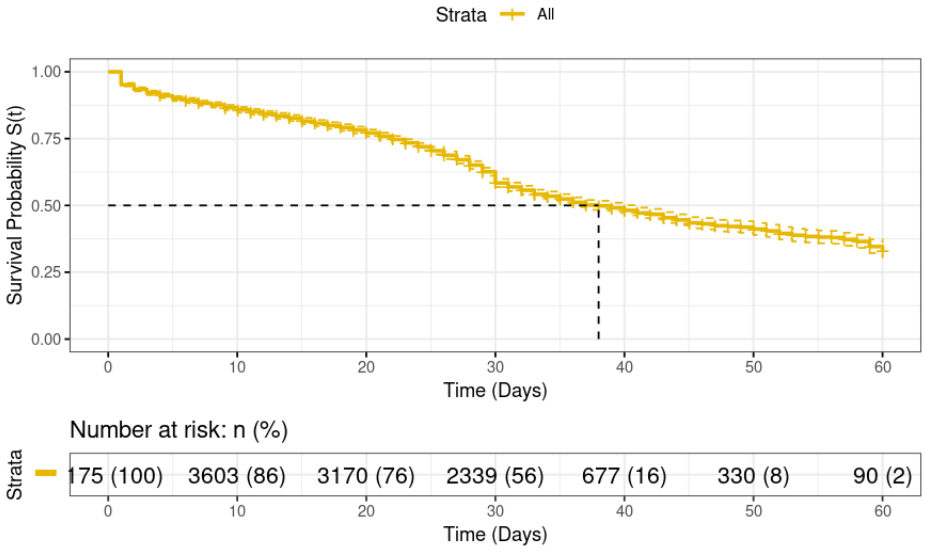

Figure 1a. Kaplan-Meier plot for most simple setup of dataset, taking no consideration of any factors besides time.
Figure 1b. Risk table with count and absolute percentage of original subjects surviving past each bloc of 10 days

Let T denote the maximum number of days in the dataset, which is 60. Expectedly, the estimated survival rate, displayed as *surv* in Figure 2, decreases over time. This trend can be symbolically expressed by $\lim_{t \to T} surv = 0$, assuming that a smooth differentiable function models the points plotted in Figure 1.

Description: df [60 x 7]

time	n.risk	n.event	n.censor	surv	upper	lower
31	1099	28	55	0.5690840	0.5852210	0.5533921
32	1016	22	36	0.5567614	0.5733845	0.5406202
33	958	26	21	0.5416509	0.5588521	0.5249792
34	911	12	19	0.5345161	0.5519805	0.5176043
35	880	17	72	0.5241902	0.5420271	0.5069403
36	791	19	20	0.5115991	0.5299292	0.4939030
37	752	15	13	0.5013943	0.5201108	0.4833513
38	724	4	8	0.4986242	0.5174442	0.4804886
39	712	11	24	0.4909207	0.5100237	0.4725332
40	677	13	73	0.4814939	0.5009450	0.4627980

31-40 of 60 rows Previous 1 2 3 4 5 6 Next

Figure 2. Sample subset of table containing KM model statistics for each unique time point.

However, the current model does not yield too much utility by using time as the only factor. Although KM cannot utilize extra factors in prediction, multiple survival curves can be created simultaneously based on levels of one or more of these factors. If M factors besides time are present with corresponding number of levels $\{m_1, m_2, \dots, m_M\}$, then there can be a maximum of $m_1 m_2 \cdots m_M$ factor-level combinations. For each subset of D with a specific M-element feature vector, a separate survival curve can be plotted. This can be visualized for D below.

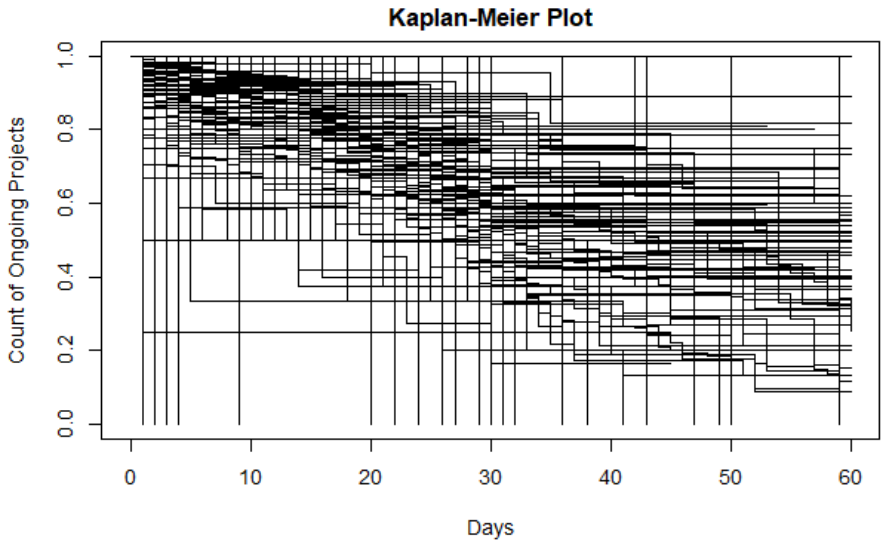

Figure 3. Kaplan-Meier plot for survival curves along all possible factor-level combinations in *D*.

To reduce the complexity of the model, dropping or selecting a subset of factors is ideal. Because more than one model for comparison cannot take in continuous factors, only columns with factor type are considered. Eventually, two factors were targeted, namely "facebook_connected" and "has_video". These factors were part of the original kickstarter dataset, while most other factors were generated or otherwise manipulated by the research team that sourced the dataset. Therefore, this subset feature selection was also driven by the choice for originality of data. The survival curves for these four (4) factor-level combinations is plotted in Figure 4.

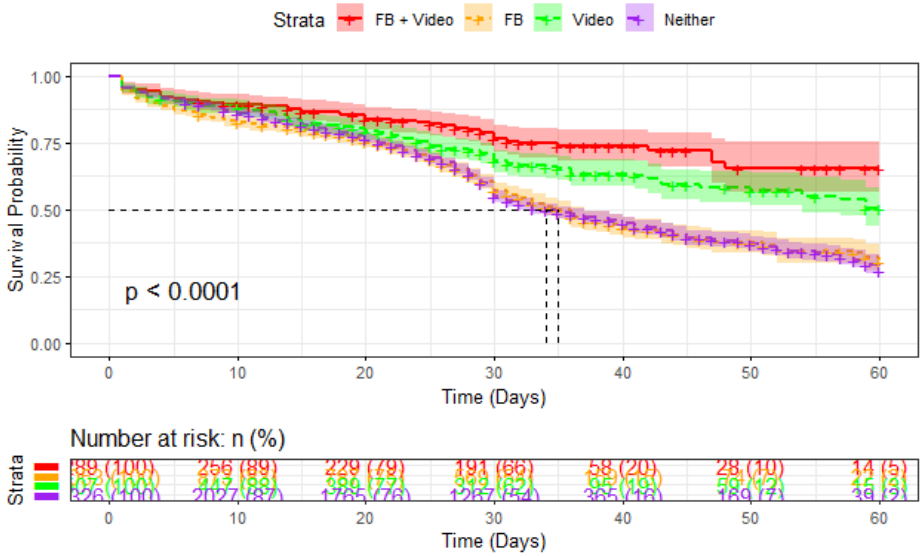

Figure 4. Kaplan-Meier plot for survival curves along all possible factor-level combinations in *D*.

Preliminaries

Definitions and brief introductions to concepts necessary to interpret the experimental results are given. Such information either cover the strategy behind machine learning models compared or the metrics that evaluate their predictive performance.

Machine Learning Models

rpart. *Regression Trees* leverage the ensemble learning advantage of multiple decision trees to regress a survival probability based on time and any factors automatically chosen. The selection of factors occur as ranked by the discriminatory power, using *Gini index*, of each factor at each stage of building the model. When building each decision node of a tree, the remaining factor with the highest Gini index becomes the deciding factor.

ctree. The *conditional inference tree* is essentially a decision with early stopping criteria. This behavior seeks to reduce bias in the model from feature selection (Horton et al., 2006).

xgboost. *Extreme gradient boost*, or xgboost, achieves a robust and effective model for classification by combining the decisions from several weak learners and concluding a final decision on the classification problem (Chen et al., 2016).

Scoring Metrics

c-index. The *concordance index*, or c-index for short, computes the proportion of all possible pairs of subjects that are concordant in time-to-event (Raykar et al., 2009). Given any two instances denoted by their respective indexes, *i* and *j* respectively, the pair is *concordant* if instance *i* has a lower *risk score* than instance *j* while also having a longer time-to-event. The risk score is computed by the estimate of the hazard function $\hat{h}(t_i) = -\frac{d}{dt}\log \hat{S}(t_i, \vec{x}_i)$. From intuition, a lower risk score should coincide with a later time-to-event when compared to a relatively higher risk score. The c-index is computed by the formula below.

$$c = \frac{\#\ concordant\ pairs}{\#\ concordant\ pairs + \#\ discordant\ pairs} = \frac{\Sigma_{i \neq j} 1_{\hat{h}(t_i) < \hat{h}(t_j)} 1_{T_i > T_j}}{\Sigma_{i \neq j} 1_{T_i > T_j}}$$

Here, t_i and t_j denote the specific time points for instances *i* and *j*, and T_i and T_j denote the true time-to-event time points for the pair. Note that the KM model would not have a c-index reported in the results, as KM cannot consider a feature vector for the hazard function.

Integrated Brier Score. Some models for binary classification make the decision for the predicted value by comparing two probabilities for either class. The higher probability determines which class becomes the predicted value for that specific test instance. In survival analysis, such a model tries to predict whether an instance would not have been affected by an event at some time *t*. The *Brier score*, also known as the *graf score*, at a *t* measures the squared difference between the probability of arriving at the prediction of the model and a 0/1 indicator for the ground truth (Square). The indicator yields 1 when the test instance has experienced an event after *t*. The following equation computes the score, which is simply the sum of the mentioned squared differences over all *N* test instances.

$$BS = \frac{1}{N}\Sigma_{i=1}^{N}\left(1_{T_i>t} - \hat{S}(t,\vec{x}_i)\right)^2$$

Here, *N* is the size of the test set, T_i is the true time-to-event for the i^{th} test instance, t is a specific time point, and $\vec{x}$ is a feature vector for the i^{th} test instance. Note that the summand is relatively large when $1_{T_i>t}$ and $\hat{S}(t,\vec{x}_i)$ disagree. For example, if the time-to-event T_i actually comes in the future ($1_{T_i>t} = 1$) but the model predicts that instance *i* will fail by time *t* already ($\hat{S}(t,\vec{x}_i)$ probability is close to 0), the summand will be close to 1. Similarly, if T_i already passed at time *t* ($1_{T_i>t} = 0$) but the model predicts that instance *i* has survived at *t* ($\hat{S}(t,\vec{x}_i)$ probability is close to 1), then the summand is again close to 1. A lesser value for *BS* generally indicates better *calibration* (Square 2022), which states how close the predicted values follow the true distribution of the survival curve created with the whole dataset in consideration. Note that the score reported from the study is the *integrated Brier Score*, which replaces the summation with an integral after treating the survival curve as a smooth line. The formula is given below.

$$\mathrm{IBS(T)} = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} BS(t)\,dt$$

Methodology

The study generally attempts to experiment each model on the same dataset as consistent as possible. Due to inherent differences between how each model operates, some nuances in dataset preparation are further explained. Across all models, the experimental procedure can be divided into two (2) logical portions: cleaning data and evaluating model.

Cleaning

Initially, the *kickstarter* dataset is accessed in CSV format. R Studio loaded in the dataset as a dataframe.

1. Each column with unique values that can form a consecutive integer range were re-formatted as factors.

2. Besides the first two columns which denote time and event status, any columns remaining not of type factor were dropped.

3. The dataset becomes further reduced by selecting factors present in the very original source of the dataset and discarding any factors that were tampered by other research.

4. For certain models, including Cox PH, contrasts between factor levels were computed as part of the training process. That required the temporary dropping of any feature columns with less than two (2) factor levels.

Model Training and Evaluation

For each model, the same instances for training were used to ensure consistency. Initially, a 75%-25% split on *D* divided the dataset into the training set and testing set. The same procedure is performed on training and evaluating each model.

1. An appropriate *learner* instance for the mlr3 package was instantiated for each model.

2. On the R software side, an appropriate *SurvTask* object was instantiated with the indexes of training instances.

3. Training is performed programmatically by the learner object.

4. Post-training, the learner object predicts output by taking in the indexes for testing instances. Again, the test set remains the same for all models. Each predicted output instance contains several values, including a binary decision on whether an event had occurred by the

timepoint from the corresponding test instance and a probability of that decision. The set of predicted outcomes is stored together into the same object.

5. A score function is called on the prediction object. This evaluates the predictions based on the metrics introduced under the Preliminaries section.

Results

Table 1. C-index and IBS for traditional models in survival analysis.

Model	Concordance Index	IBS
Kaplan-Meier	0.5000	0.1914
Cox Proportional-Hazards	0.5065	0.1895

Table 2. C-index and IBS for tree-based machine learning models adapted for survival analysis.

Model	Concordance Index	IBS
rpart	0.5205	0.2746
ctree	0.8671	0.2851
xgboost	0.5204	0.1908

Conclusion

Based on the collected metrics, tree-based models outperformed the traditional models by a sizeable margin. The highest c-index from the traditional models was still lower than the lowest c-index from the tree-based models. This suggests that using decision trees, random forests, boosting, and similar ensemble strategies would bring more discriminatory power when predicting whether a given time point and set of features would have already met an event. However, the traditional models did have lower IBS scores overall. This suggests that both the KM and Cox PH models capture the true distribution of event ratios occurring across time points when compared to tree-based models. For future work, extending the study to other machine learning models would be ideal for a broader comparison of machine learning performance in survival analysis studies. Additionally, conducting an explanatory analysis on the causes of each group achieving a higher c-index and IBS score would add more understanding.

References

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 785–794. https://doi.org/10.1145/2939672.2939785

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. In *Journal of Computational and Graphical Statistics* (Vol. 15, Issue 3, pp. 651–674). https://doi.org/10.1198/106186006X133933

Li, Y., Rakesh, V., & Reddy, C. K. (2016). Project success prediction in crowdfunding environments. *WSDM 2016 - Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, 247–256. https://doi.org/10.1145/2835776.2835791

Raykar, V. C., Steck, H., Krishnapuram, B., Dehing-Oberije, C., & Lambin, P. (2009). On ranking in survival analysis: Bounds on the concordance index. *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, 1–8.

Square. (n.d.). *Welcome to pysurvival.io*. PySurvival. Retrieved April 14, 2022, from https://square.github.io/pysurvival/

Sonabend, R. (n.d.). *Probabilistic supervised learning for MLR3*. Probabilistic Supervised Learning for mlr3 •. Retrieved April 14, 2022, from https://mlr3proba.mlr-org.com/