Journal of Machine Learning Research 21 (2020) 1-6 Submitted 7/20; Revised 10/20; Published 10/20

scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn

Sebastian Pölsterl SEBASTIAN.POELSTERL@MED.UNI-MUENCHEN.DE
Artificial Intelligence in Medical Imaging (AI-Med),
Department of Child and Adolescent Psychiatry,
Ludwig-Maximilians-Universität, Munich, Germany

Editor: Andreas Mueller

Abstract

`scikit-survival` is an open-source Python package for time-to-event analysis fully compatible with `scikit-learn`. It provides implementations of many popular machine learning techniques for time-to-event analysis, including penalized Cox model, Random Survival Forest, and Survival Support Vector Machine. In addition, the library includes tools to evaluate model performance on censored time-to-event data. The documentation contains installation instructions, interactive notebooks, and a full description of the API. `scikit-survival` is distributed under the GPL-3 license with the source code and detailed instructions available at `https://github.com/sebp/scikit-survival`

Keywords: Time-to-event Analysis, Survival Analysis, Censored Data, Python

1. Introduction

In time-to-event analysis—also known as survival analysis in medicine, and reliability analysis in engineering—the objective is to learn a model from historical data to predict the future time of an event of interest. In medicine, one is often interested in prognosis, i.e., predicting the time to an adverse event (e.g. death). In engineering, studying the time until failure of a mechanical or electronic systems can help to schedule maintenance tasks. In e-commerce, companies want to know if and when a user will return to use a service. In all of these domains, learning a model is challenging, because the time of past events is only partially known. Consider a clinical study carried out over a one year period that studied survival times after treatment. In most cases, only a subset of patients will die during the study period, while others will live beyond the end of the study. To learn a predictive model of survival time, we need to consider that the exact day of death is only known for those patients that actually died during the study period, while for the remaining patients, we only know that they were alive at least until the study ended. The latter is called right censoring and occurs in all applications mentioned above.

Today, many successful ideas from machine learning have been adapted for time-to-event analysis, such as gradient boosting (Ridgeway, 1999; Hothorn et al., 2006), random forests (Ishwaran et al., 2008), and support vector machines (Van Belle et al., 2007; Evers and Messow, 2008; Pölsterl et al., 2015). It is important to note that censored data does not only affect model training, but also model evaluation, because held-out data will be subject to censoring too. Evaluation metrics range from simple rank correlation metrics (Harrell

et al., 1996; Uno et al., 2011) to time-dependent versions of the well-known mean squared error (Graf et al., 1999) and receiver operating characteristic curve (Hung and Chiang, 2010; Uno et al., 2007).

In this paper, we present `scikit-survival`, a Python library for time-to-event analysis. It provides a tight integration with `scikit-learn` (Pedregosa et al., 2011), such that pre-preprocessing and feature selection techniques within `scikit-learn` can be seamlessly combined with a model from `scikit-survival`. It provides efficient implementations of linear models, ensemble models, and survival support vector machines, as well as a range of evaluation metrics suitable for right censored time-to-event data.

2. Overview and Design

The API of `scikit-survival` is designed to be compatible with the `scikit-learn` API, such that existing tools for cross validation, feature transformation, and model selection can be used for time-to-event analysis. Each model in `scikit-survival` is a sub-class of scikit-learn's `BaseEstimator` class, which offers users a familiar API to set hyper-parameters via `set_params()`, train a model via `fit()`, and evaluate it on held-out data via `score()`. Time-to-event data is comprised of a time $t > 0$ when an event occurred and the time $c > 0$ of censoring. Since censoring and experiencing and event are mutually exclusive, it is common to define a binary event indicator $\delta = I(t \leq c) \in \{0; 1\}$ and a observable time $y = t$ (if $\delta = 1$) or $y = c$ (if $\delta = 0$). To allow integration with `scikit-learn`, which expects a single array as dependent variable, `scikit-survival` uses numpy's structured arrays, whose data type is a composition of simpler data types. The `sksurv.util.Surv` utility helps users to create such arrays from `pandas` DataFrames or individual `numpy` arrays.

The biggest difference between time-to-event analysis and traditional machine learning tasks are the semantics of predictions. Predictions in time-to-event analysis are often arbitrary risk scores of experiencing an event, and not an actual time of an event, which is the input for training such a model. Consequently, predictions are often evaluated by a measure of rank correlation between predicted risk scores and observed time points in the test data. For instance, Harrell's concordance index (Harrell et al., 1996) computes the ratio of correctly ordered (concordant) pairs to comparable pairs and is the default performance metric when calling a model's `score()` method. Models that provide time-dependent predictions in the form of survival function and cumulative hazard function, have two additional prediction methods: `predict_survival_function()`, and `predict_cumulative_hazard_function()`. Such time-dependent predictions can be evaluated on held-out data using the time-dependent Brier score (Graf et al., 1999).

3. Development

Where possible, `scikit-survival` uses scikit-learn's efficient implementations to fit models. For instance, `RandomSurvivalForest` leverages scikit-learn's Cython implementation to fit tree-based estimators by introducing the log-rank node-splitting criterion for right censored data. Therefore, the computationally expensive training step usually runs natively, which leads to high efficiency. To ensure high quality code, all implementations adhere to the PEP8 code style, have inline documentation, and are accompanied by unit tests that are executed

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sksurv.datasets import load_whas500
from sksurv.linear_model import CoxPHSurvivalAnalysis
from sksurv.preprocessing import OneHotEncoder

# load example data
data_x, data_y = load_whas500()
# split the data
X_train, X_test, y_train, y_test = train_test_split(
    data_x, data_y, test_size=50, random_state=2020)

# combine feature transform and Cox model
pipeline = make_pipeline(
    OneHotEncoder(), CoxPHSurvivalAnalysis())
# fit the model
pipeline.fit(X_train, y_train)
# compute concordance index on held-out data
c_index = pipeline.score(X_test, y_test)

# plot estimated survival functions
surv_fns = pipeline.predict_survival_function(X_test)
time_points = np.arange(1, 1000)
for surv_func in surv_fns:
    plt.step(time_points, surv_func(time_points),
             where="post")
plt.ylabel("probability of survival $\hat{S}(t)$")
plt.xlabel("time $t$")
plt.title("concordance index = %.3f" % c_index)
plt.show()
```

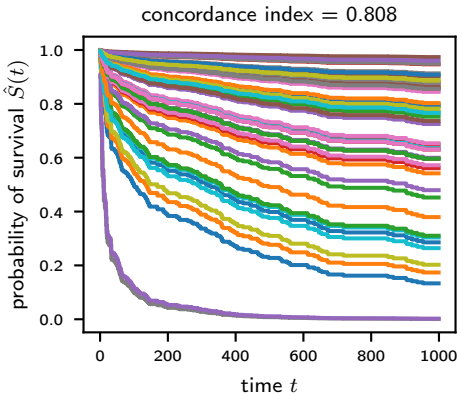

Figure 1: Example of using `scikit-survival` (left) and its output (right).

on all supported platforms by our continuous integration workflow. As a result, units tests cover 98% of our code, and code quality is rated A by Codacy.[1]

4. Installation and Usage

`scikit-survival` is hosted on GitHub[2], interactive tutorials in the form of notebooks and API documentation are available on the *Read the Docs* platform.[3] Pre-compiled packages for Python 3.6 and above are available for Linux, macOS, and Windows, and can be obtained via Anaconda using `conda install -c sebp scikit-survival`. Figure 1 shows a basic example of fitting and evaluating a model in `scikit-survival`. More elaborate examples can be found in the online documentation.

1. `https://www.codacy.com/app/sebp/scikit-survival`
2. `https://github.com/sebp/scikit-survival`
3. `https://scikit-survival.readthedocs.io/`

Task	Model	sksurv	lifelines	statsmodels	pycox
Survival Function Estimation	Non-parametric	✓	✓	✓	✓
	Parametric	✗	✓	✗	✗
CHF Estimation	Non-parametric	✓	✓	✗	✗
	Parametric	✗	✓	✗	✗
Linear Regression	Cox	✓	✓	✓	✓
	Cox + Elastic-Net	✓	✓	✓	✗
	Log-normal AFT	✓	✓	✗	✗
	Log-logistic AFT	✗	✓	✗	✗
	Weibull AFT	✗	✓	✗	✗
	Piecewise exponential	✗	✓	✗	✓
Non-linear Regression	Aalen	✗	✓	✗	✗
	Gradient Boosted AFT	✓	✗	✗	✗
	Gradient Boosted Cox	✓	✗	✗	✗
	Heterogenous Ensemble	✓	✗	✗	✗
	NN (Grouped survival times)	✗	✗	✗	✓
	NN (Proportional hazards)	✗	✗	✗	✓
	NN (Piecewise exponential)	✗	✗	✗	✓
	Random Survival Forest	✓	✗	✗	✗
	Survival SVM	✓	✗	✗	✗
	Survival Tree	✓	✗	✗	✗
Evaluation	Brier Score	✓	✗	✗	✓
	Concordance Index	✓	✓	✗	✓
	Time-dependent ROC	✓	✗	✗	✗

Table 1: Availability of methods. AFT: Accelerated Failure Time. CHF: Cumulative Hazard Function. NN: Neural Network. SVM: Support Vector Machine.

5. Comparison to Related Software

The R language currently offers many packages for time-to-event analysis. Usually, each package focuses on a specific type of model, such as tree-based models, and API and code quality can vary widely across packages. Options for the Python scientific computing stack are currently limited. `statsmodels` (Seabold and Perktold, 2010) only has basic support; it includes Cox's proportional hazards models (Cox, 1972), the Kaplan-Meier estimator (Kaplan and Meier, 1958), and the log-rank test (Peto and Peto, 1972). Its API is not compatible with `scikit-learn`. The `lifelines` package (Davidson-Pilon et al., 2020) includes alternative implementations of those and additionally includes parametric estimators of the survival function, additive, semi-parametric and parametric regression models. Recent versions also include an experimental compatibility layer for integration with the `scikit-learn` API. `pycox` (Kvamme et al., 2019) focuses on neural networks and provides losses for proportional hazards, grouped survival time, and piecewise exponential models that can be minimized by stochastic gradient descent. In contrast to the above, `scikit-survival` is designed to be fully compatible with the `scikit-learn` API, includes traditional linear models, modern machine learning models, and a range of evaluation metrics. Table 1 presents a detailed comparison.

6. Conclusion

`scikit-survival` is a Python package that provides implementations of popular machine learning models and evaluation metrics for time-to-event analysis. Thanks to its compatibility with the `scikit-learn` API, users can utilize existing tools for cross-validation and model selection to create powerful analysis pipelines.

References

D. R. Cox. Regression models and life tables. *Journal of the Royal Statistical Society: Series B*, 34:187–220, 1972.

C. Davidson-Pilon, J. Kalderstam, N. Jacobson, sean reed, B. Kuhn, P. Zivich, M. Williamson, AbdealiJK, D. Datta, A. Fiore-Gartland, A. Parij, D. WIlson, Gabriel, L. Moneda, K. Stark, A. Moncada-Torres, H. Gadgil, Jona, K. Singaravelan, L. Besson, M. S. Peña, S. Anton, A. Klintberg, J. Noorbakhsh, M. Begun, R. Kumar, S. Hussey, D. Golland, jlim13, and A. Flaxman. *CamDavidsonPilon/lifelines*. Zenodo, 2020. doi: 10.5281/zenodo.805993.

L. Evers and C.-M. Messow. Sparse kernel methods for high-dimensional survival data. *Bioinformatics*, 24(14):1632–1638, 2008. doi: 10.1093/bioinformatics/btn253.

E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, 18(17-18):2529–2545, 1999. doi: 10.1002/(SICI)1097-0258(19990915/30)18:17/18<2529::AID-SIM274>3.0.CO;2-5.

F. E. Harrell, K. L. Lee, and D. B. Mark. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, 15(4):361–387, 1996. doi: 10.1002/(SICI)1097-0258(19960229)15:4<361::AID-SIM168>3.0.CO;2-4.

T. Hothorn, P. Bühlmann, S. Dudoit, A. Molinaro, and M. J. van der Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 2006. doi: 10.1093/biostatistics/kxj011.

H. Hung and C. T. Chiang. Estimation methods for time-dependent AUC models with survival data. *Canadian Journal of Statistics*, 38(1):8–26, 2010. doi: 10.1002/cjs.10046.

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 2008. doi: 10.1214/08-aoas169.

E. L. Kaplan and P. Meier. Nonparametric Estimation from Incomplete Observations. *Journal of the American Statistical Association*, 53:457–481, 1958. doi: 10.2307/2281868.

H. Kvamme, Ørnulf Borgan, and I. Scheel. Time-to-Event Prediction with Neural Networks and Cox Regression. *Journal of Machine Learning Research*, 20(129):1–30, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

R. Peto and J. Peto. Asymptotically Efficient Rank Invariant Procedures. *Journal of the Royal Statistical Society: Series A*, 135(2):185–207, 1972. doi: 10.2307/2344317.

S. Pölsterl, N. Navab, and A. Katouzian. Fast Training of Support Vector Machines for Survival Analysis. In A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, and C. Soares, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 243–259, 2015. doi: 10.1007/978-3-319-23525-7_15.

G. Ridgeway. The state of boosting. *Computing Science and Statistics*, pages 172–181, 1999.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

H. Uno, T. Cai, L. Tian, and L. J. Wei. Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models. *Journal of the American Statistical Association*, 102: 527–537, 2007. doi: 10.1198/016214507000000149.

H. Uno, T. Cai, M. J. Pencina, R. B. D'Agostino, and L. J. Wei. On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in Medicine*, 30(10):1105–1117, 2011. doi: 10.1002/sim.4154.

V. Van Belle, K. Pelckmans, J. A. K. Suykens, and S. Van Huffel. Support Vector Machines for Survival Analysis. In *Proc. of the 3rd International Conference on Computational Intelligence in Medicine and Healthcare*, pages 1–8, 2007.