`auton-survival`: an Open-Source Package for Regression, Counterfactual Estimation, Evaluation and Phenotyping with Censored Time-to-Event Data

Chirag Nagpal, Willa Potosnak and Artur Dubrawski

{CHIRAGN, WPOTOSNA, AWD}@ANDREW.CMU.EDU

Auton Lab, School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, United States

autonlab.github.io/auton-survival

Abstract

Applications of machine learning in healthcare often require working with time-to-event prediction tasks including prognostication of an adverse event, re-hospitalization, and mortality. Such outcomes are typically subject to censoring due to loss of follow up. Standard machine learning methods cannot be applied in a straightforward manner to datasets with censored outcomes. In this paper, we present `auton-survival`, an open-source repository of tools to streamline working with censored time-to-event or survival data. `auton-survival` includes tools for survival regression, adjustment in the presence of domain shift, counterfactual estimation, phenotyping for risk stratification, evaluation, as well as estimation of treatment effects. Through real world case studies employing a large subset of the SEER oncology incidence data, we demonstrate the ability of `auton-survival` to rapidly support data scientists in answering complex health and epidemiological questions.

Introduction

Machine learning is being increasingly applied to address challenges across multiple areas of healthcare. Healthcare data is inherently complex and thus offers a multitude of challenges to traditional machine learning. The advent of deep neural networks has especially catalyzed interest in the use of machine learning for healthcare as these models can be used to learn nonlinear representations of complex clinical data.

Modern machine learning approaches inherently involve reasoning and inference in terms of binary or categorical outcomes. In reality, healthcare outcomes often are continuous time-to-events, such as mortality, stroke, onset of cancer, and re-hospitalization. The challenges of working with time-to-event data are further compounded by the fact that data typically includes individuals whose outcomes are missing, or *censored* due to loss of follow up. Standard classification and regression approaches do not provide a straightforward way to deal with such censored data.

Bio-statistics and medical informatics literature has extensively dealt in methodology to deal with censored time-to-event outcomes. However, such methodology is relatively understudied in modern machine learning. As a result, there is limited support in terms of robust, reproducible software that is equipped to handle censored data. In this paper, we present the package `auton-survival`, a comprehensive `python` repository of user-friendly utilities for application of machine learning in the presence of censored time-to-event data.

Generalizable Insights about Machine Learning in the Context of Healthcare:
Healthcare research is replete with problems involving assessment of time-to-events. From
critical care, oncology, and epidemiology to cardio-vascular, mental, and public health,
problems involving analysis of time-to-event outcomes are ubiquitous. Examples include
mortality, hospital discharge, and onset of an adverse clinical conditions such as stroke and
cancer. The analysis of such time-to-event data, especially when subjected to censoring,
requires a specialized set of tools different from standard classification or regression.

Technical Significance: Current popular machine learning frameworks involve modelling
problems as classification or regression. Applying these existing frameworks to healthcare
problems often involves discretizing time-to-event outcomes as binary classification. However,
this approach neglects temporal context which could result in models with misestimation
and poor generalization. While bio-statistics literature has extensively dealt with methodol-
ogy and corresponding software for censored time-to-event outcomes, support support for
survival outcomes through modern representation and machine learning techniques is limited.
`auton-survival` is one step towards providing support to work with time-to-event data and
includes an exclusive suite of workflows that allow for a multitude of experiments from data
pre-processing and regression modelling to model evaluation. Additionally, `auton-survival`
uses an API similar to `scikit-learn` (Pedregosa et al., 2011), making adoption easy for
users already familiar with machine learning in `Python`.

`auton-survival` includes thorough documentation of utilities as well as example code
notebooks to facilitate rapid prototyping. `auton-survival` is open source and hosted on
GitHub[1] to promote widespread use of the package for reproducible machine learning for
healthcare research.

Related Work and Contributions

Modelling of censored time-to-event outcomes is recently gaining attention in the machine
learning community. There have been successful attempts to incorporate deep non-linear
representations in the classic Cox Proportional Hazards model (Faraggi and Simon, 1995;
Katzman et al., 2018). Considerable attention has been given to the problem of easing the
strong restrictive assumptions of the proportional hazards model involving discrete time (Yu
et al., 2011; Lee et al., 2018) as well as parametric (Ranganath et al., 2016; Nagpal et al.,
2021b) and adversarial (Chapfuwa et al., 2018) approaches.

There have been other attempts at reproducible machine learning pipelines with `Python`:
the `lifelines` (Davidson-Pilon, 2019) and `scikit-survival` (Pölsterl, 2020) packages offer
support for classical standard survival regression methods and `pycox` (Kvamme et al., 2019)
attempts to streamline the application of `torch` based deep learning models with simpler
APIs. A broad comparison of the extended functionalities offered by `auton-survival` is in
Appendix A.

`auton-survival` builds upon many of the software design choices of existing `Python`
packages for machine learning and survival analysis (Davidson-Pilon, 2019; Paszke et al.,
2019; Pölsterl, 2020). It offers functionality for rapid experimentation with multiple classes
of survival regression models and corresponding metrics to evaluate model discriminative

---

1. `github.com/autonlab/auton-survival`

capability and calibration. Additionally, `auton-survival` uniquely offers easy to use APIs for counterfactual and treatment effect estimation as well as subgroup discovery, among other utilities, to solve the following real world problems involving censored time-to-events:

Counterfactual and Treatment Effect Estimation: Clinical decision support often requires reasoning about *what if*? scenarios. In situations where outcomes maybe confounded, such inference requires estimating counterfactuals by adjusting for confounding.

Survival Regression with Time-Varying Covariates: Real world health data often consists of multiple time-dependent observations per individual, or time-varying covariates. `auton-survival` has support for auto-regressive deep learning models that allow learning temporal dependencies when estimating time-to-event outcome.

Subgroup and Phenotype Discovery and Evaluation: Inherent heterogeneity in patient populations results in differential event incidence rates conditioned on covariates and interventions. Identifying patient subgroups with differential risk can provide insight into best practices that benefit such phenogroups.

Contents

1. Time-to-event or Survival Regression

Throughout this paper, we will work with a dataset of right censored instances $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{t}_i, \delta_i)\}_{i=1}^n$ where $\boldsymbol{x}_i$ is the set of covariates of an individual. $\boldsymbol{t}_i$ is the time to event or censoring $\mathbb{R}^+$ as indicated by the indicator $\delta_i \in \{0, 1\}$. Time-to-event or survival estimation problem thus reduces to estimating the conditional distribution of survival notated as

$$\mathbb{E}[\mathbf{1}\{T > \boldsymbol{t}\}|X = \boldsymbol{x}]$$
$$= \mathbb{P}(T > \boldsymbol{t}|X = \boldsymbol{x})$$
$$= 1 - \mathbb{P}(T \leq \boldsymbol{t}|X = \boldsymbol{x}). \quad (1)$$
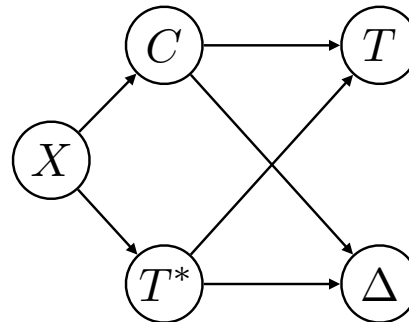

Figure 1: Conditional Independence of the True Time-to-Event T^* and the censoring times C. Only X, T and Δ are observed.

Here, T refers to the distribution of the censored survival time $T = \min(T^*, C)$, where T^* is the distribution of the true time-to-event and C is the distribution of the censoring time (Figure 1). Δ is the distribution of the censoring indicator $\Delta = \mathbf{1}\{T^* < C\}$. Typically we do not observe the true event times for individuals lost to follow up as in Figure 2.
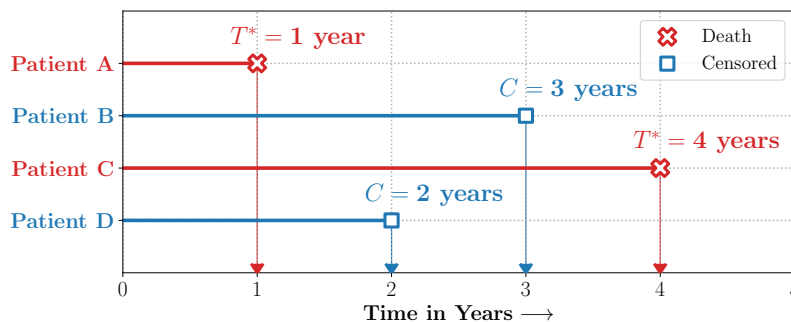

Figure 2: **Censoring and Time-to-Event Predictions**: **Patients A** and **C** died **1** and **4 years** from entry into the study, whereas **Patients B** and **D** exited the study without experiencing death (were lost to follow up) at **2** and **3 years** from entry in the study. *Time-to-Event* or *Survival Regression* thus involves estimates that are adjusted for individuals whose outcomes were censored.

For these individuals, we observe the censored survival time, T and an indicator of if they were censored, $\Delta = T < C$. Assuming conditional independence between T and C ie. $T \perp C|X$ allows identification of the distribution of $\mathbb{P}(T|X)$.

For the censored individuals, we maximize the probability corresponding to the survival function. The likelihood, $\ell(\cdot)$ under censoring is thus given as

$$\boldsymbol{\ell}(\{\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{\delta}\}) \propto \mathbb{P}(T = \boldsymbol{t}|X = \boldsymbol{x})^\delta \mathbb{P}(T > \boldsymbol{t}|X = \boldsymbol{x})^{1-\delta}. \quad (2)$$

Often in survival analysis literature likelihoods are expressed in terms of instantaneous hazard rates $\boldsymbol{\lambda}(t)$. The instantaneous hazard maybe defined as the event rate at a time t,
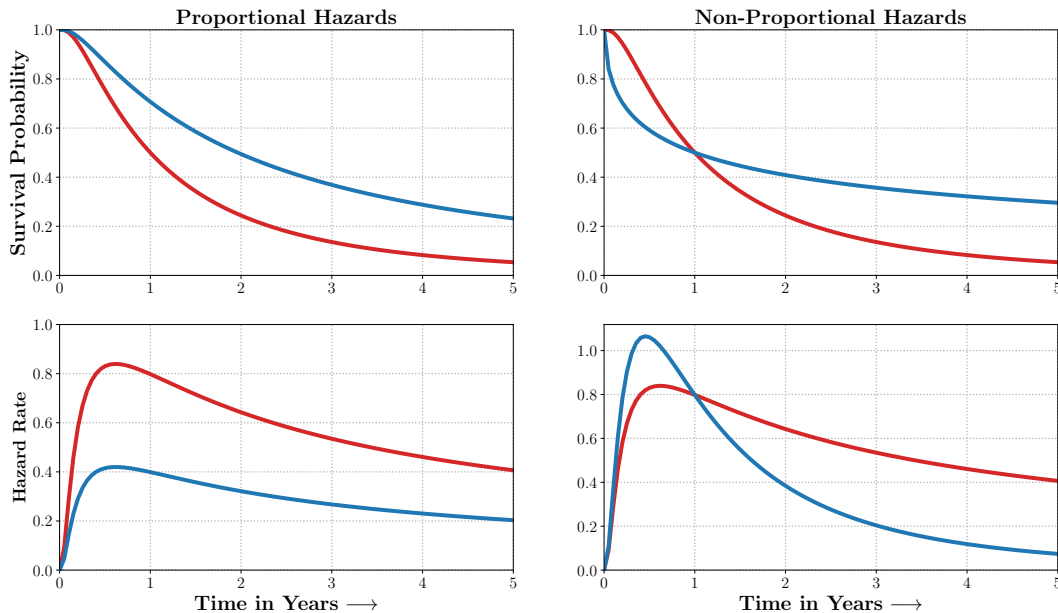
Figure 3: **Non-Proportional Hazards**: When the Proportional Hazards assumptions are satisfied, the Survival Curves and their corresponding Hazard Rates dominate each other and do not intersect. In many real world scenarios however, the survival curves. `auton-survival` includes flexible estimators of Time-to-Events in the presence of non-proportional hazards.

conditional on survival $(T > t)$ till that time. Thus,

$$\boldsymbol{\lambda}(t) = \lim_{\mathrm{d}t \to 0} \frac{\mathbb{P}(t \leq T < t + \mathrm{d}t)}{\mathrm{d}t \cdot S(t)} = \frac{\mathbb{P}(T = t)}{\boldsymbol{S}(t)} \quad \text{or,} \quad \boldsymbol{\lambda}(t) = -\frac{\mathrm{d}}{\mathrm{d}t} \log \boldsymbol{S}(t). \tag{3}$$

Now reasoning in terms of hazard rates, we can rewrite Equation 2 equivalently as

$$\boldsymbol{\ell}(\{\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{\delta}\}) \propto \boldsymbol{\lambda}(\boldsymbol{t}|X = \boldsymbol{x})^{\delta} \boldsymbol{S}(\boldsymbol{t}|X = \boldsymbol{x}). \tag{4}$$

Broadly the popular approaches for maximizing the likelihood above are classified as,

Parametric Assumes the distribution of the time-to-event $\mathbb{P}(T|X = \boldsymbol{x})$ is parametric like Weibull or Log-Normal. Examples includes the popular Accelerated Failure Time model.

Non-Parametric Involves learning kernels or similarity functions of the input covariates followed by a non-parametric (Kaplan-Meier or Nelson-Aalen) estimation of the survival rate weighted with the learnt kernel.

Semi-Parametric the Cox Proportional Hazards model and its extensions arguably, remain the most popular approaches and are classified as *semi-parametric*. The Cox model involves a two step estimation where the feature interactions are learnt through a parametric model followed by non-parametric estimation of the base survival (hazard) rate.

1.1. Fitting Survival Estimators

auton-survival includes the standard Cox Proportional Hazards (CPH) (Cox, 1972) to estimate time-to-event regression models. auton-survival allows the use of deep representation learning approaches to learn the Cox model as in Faraggi and Simon (1995); Katzman et al. (2018). Note that the standard Cox model is sensitive to the strong assumptions of *Proportional Hazards* which maybe mis-specified in real world scenarios (Figure 3). auton-survival thus includes deep latent variable survival regression models *Deep Cox Mixtures* (DCM) and *Deep Survival Machines* (DSM) (Nagpal et al., 2021b,c) that ease this restrictive assumption by modeling the conditional time-to-event distributions as mixtures of fixed size.

We demonstrate the use of auton-survival to predict long term mortality for 9,105 critically ill hospitalized patients from the SUPPORT (Study to Understand Prognoses Preferences Outcomes and Risks of Treatment) (Knaus et al., 1995; Connors et al., 1995) study; consisting of demographics, physiological measurements and outcomes followed up over a 5 year period.

```python
from auton_survival import datasets, preprocessing, models
# Load the SUPPORT Dataset
outcomes, features = datasets.load_dataset("SUPPORT")
# Preprocess (Impute and Scale) the features
features = preprocessing.Preprocessor().fit_transform(features)
# Train a Deep Cox Proportional Hazards (DCPH) model
model = models.cph.DeepCoxPH(layers=[100])
model.fit(features, outcomes.time, outcomes.event)
# Predict risk at specific time horizons.
predictions = model.predict_risk(features, t=[8, 12, 16])
```

auton-survival also provides the convenient, SurvivalModel class that enables rapid experimentation via a consistent API by wrapping multiple regression estimators. In addition to the models above, SurvivalModel class also includes Random Survival Forests (RSF) (Ishwaran et al., 2008) a popular non-parametric survival model.

```python
from auton_survival import estimators                          [Demo Notebook]
# Train a Deep Survival Machines model using the SurvivalModel class.
model = estimators.SurvivalModel(model='dsm')
model.fit(features, outcomes)
# Predict risk at time horizons.
predictions = model.predict_risk(features, times=[8, 12, 16])
```

Additionally, the SurvivalRegressionCV class can be used to optimize survival regression models in a K-fold cross validation fashion over a user specified hyperparameter grid. Model selection is performed by selecting the model that minimizes the Integrated Brier Score.

```python
from auton_survival.experiments import SurvivalRegressionCV    [Demo Notebook]
# Define the Hyperparameter grid to perform Cross Validation
hyperparam_grid = {'n_estimators' : [50, 100], 'max_depth' : [3, 5],
                   'max_features' : ['sqrt', 'log2']}
# Train a RSF model with cross-validation using the SurvivalRegressionCV class
model = SurvivalRegressionCV(model='rsf', cv_folds=5, hyperparam_grid=hyperparam_grid)
model.fit(features, outcomes)
```

1.2. Importance Weighting

Frequently in survival regression, we need to perform inference over a test dataset subject to distribution shift. Distribution shift can arise in multiple ways and affect the generalizability of a model. For simplicity, consider the covariate shift problem where, $\mathbb{P}_{\text{train}}(T, \Delta|X) = \mathbb{P}_{\text{test}}(T, \Delta|X)$ but $\mathbb{P}_{\text{train}}(X) \neq \mathbb{P}_{\text{test}}(X)$. auton-survival allows a flexible API to adjust for distribution shift involving Importance Weighted Empirical Risk Minimization (IWERM) (Sugiyama et al., 2007; Shimodaira, 2000) by weighted resampling of the training data with replacement at training time.

$$\underset{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{\delta})\sim\mathcal{D}_{\text{test}}}{\mathbb{E}}[\mathcal{L}(\boldsymbol{x},\boldsymbol{t},\boldsymbol{\delta})] = \underset{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{\delta})\sim\mathcal{D}_{\text{test}}}{\mathbb{E}}\left[\ln\left(\boldsymbol{\lambda}(t|X=\boldsymbol{x})^{\delta}\boldsymbol{S}(t|X=\boldsymbol{x})\right)\right] \tag{5}$$

$$= \int_{\boldsymbol{x}\in\mathcal{X}}\int_{\boldsymbol{t}\in\mathcal{T}}\int_{\delta\in\{0,1\}}\ln\left(\boldsymbol{\lambda}(t|X=\boldsymbol{x})^{\delta}\boldsymbol{S}(t|X=\boldsymbol{x})\right)\mathbb{P}_{\text{test}}(\boldsymbol{x},\boldsymbol{t},\delta) \tag{6}$$

$$= \underset{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{\delta})\sim\mathcal{D}_{\text{train}}}{\mathbb{E}}\left[\boldsymbol{w}(\boldsymbol{x})\cdot\ln\left(\boldsymbol{\lambda}(t|X=\boldsymbol{x})^{\delta}\boldsymbol{S}(t|X=\boldsymbol{x})\right)\right], \tag{7}$$

here, $\boldsymbol{w}(\boldsymbol{x}) \propto \dfrac{\mathbb{P}_{\text{test}}(\boldsymbol{x})}{\mathbb{P}_{\text{train}}(\boldsymbol{x})}$ are the estimated importance weights.

```
1  # Estimate Importance Weights with Logistic Regression
2  from sklearn.linear_model import LogisticRegression
3  p_target = LogisticRegression().fit(features, domains).predict_proba(features_source)
4  imp_weights = p_target/(1-p_target) # Propensity Weighting
5  # Train the Survival Regression Model
6  from auton_survival.estimators import SurvivalEstimator
7  model = SurvivalEstimator("dcph", layers=[100]) # Cox PH Model with 1 Hidden Layer
8  model.fit(features=features_source, outcomes=outcomes_source, weights=imp_weights)
```

Cautionary Note (Censoring and Distribution Shift)

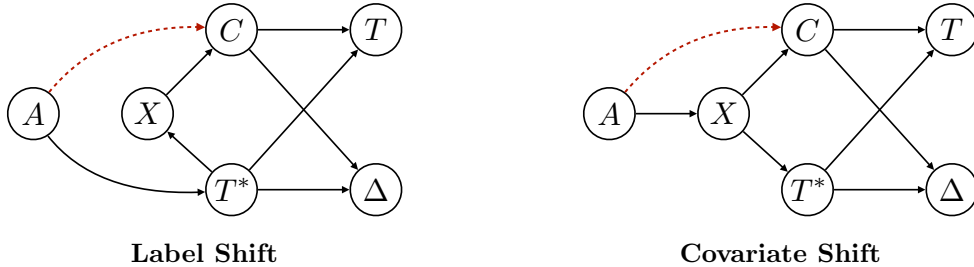Label Shift   Covariate Shift

Figure 4: *Prior probability* and *Covariate Shift* in Time-to-Event (Survival) Regression. In situations where $A \rightarrow C$ the censoring rates, C are dependent on the domain A, it may not be possible adjust for domain shift with standard importance weighting. Readers are thus advised to exercise caution in scenarios where shifts result in such complex interactions.

For the above covariate shift adjustment to work, the conditional distribution of the censoring times must be the same across the source and target domains. Additionally, is common to use importance weighting for adjustment in situations where $\mathbb{P}_{\text{train}}(T) \neq \mathbb{P}_{\text{test}}(T)$ This is commonly referred to as *label shift* or *prior probability shift* in Machine Learning (Figure 4). As in the case of covariate shift adjustment, this requires additional assumptions

on the censoring distributions which might be violated in real-world situations. Readers are thus recommended to exercise appropriate caution when reasoning about whether importance weighting is an appropriate approach to adjust for distribution shift. A thorough discussion of the implications of this is beyond the current scope of this manuscript.

1.3. Counterfactual Survival Regression

Survival outcomes are often used to answer '*what if?*' questions requiring inference of counterfactuals. We notate an intervention with an indicator $\boldsymbol{a} \in \{0, 1\}$ and assume **Strong Ignorability (Unconfoundedness)** that the potential outcomes $T(1)$ and $T(0)$ are independent of the treatment assignment, $(A = \boldsymbol{a})$ conditioned on the set of confounders $(X = \boldsymbol{x})$ (Rubin, 2005). The estimated time-to-event outcome under intervention $(A = \boldsymbol{a})$ is

$$\widehat{\boldsymbol{S}}\big(t|\mathrm{do}(A = \boldsymbol{a})\big) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}\big[\widehat{\mathbb{E}}[\mathbf{1}\{T > t\}|X = \boldsymbol{x}, A = \boldsymbol{a}]\big]. \tag{8}$$

Note that $\widehat{\mathbb{E}}[\mathbf{1}\{T > t\}|X = \boldsymbol{x}, A = \boldsymbol{a}]$ is just a conditional estimator of survival learnt on the population under intervention $\boldsymbol{a}$. In practice thus, the counterfactual survival regression involves fitting separate regression models on the treated and control populations. auton-survival allows learning counterfactual models with K-fold cross-validation using the CounterfactualSurvivalCV class.

```
1 from auton_survival.experiments import CounterFactualSurvivalRegressionCV
2 grid = {'layers' : [[], [100]], 'learning_rate' : [1e-3, 1e-4]} # Hyperparameter Grid
3 # Train a counterfactual Cox model with cross-validation.
4 model = CounterfactualSurvivalRegressionCV('dcph', 5, hyperparam_grid=grid)
5 model.fit(features, outcomes)
```

1.4. Time-Varying Survival Regression

Additionally auton-survival also includes *time-varying* implementations of Deep Survival Machines (Nagpal et al., 2021a) and Deep Cox Proportional Hazards model (Lee et al., 2021) thats involves the use of RNNs, LSTMs or GRUs (Chung et al., 2014; Hochreiter and Schmidhuber, 1997). Figure 5 presents time varying survival regression in auton-survival. For an individual with
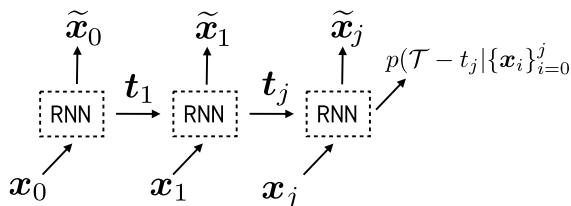

Figure 5: Time Varying Survival Regression

time-to-event $\mathcal{T}_i$, we observe covariates $\boldsymbol{x}_i^j$ at multiple time points t_i^j. At each time-step, j we estimate the distribution of the remaining time-to-event $T_i - t_i^j$. The representations of the input covariates $\widetilde{\boldsymbol{x}}_i^{(j+1)}$ at time-step $(j + 1)$ are functions of the covariates, $\boldsymbol{x}_i^{(j+1)}$ and the representation of the preceeding time-step $\widetilde{\boldsymbol{x}}_i^j$. auton-survival automatically handles appropriate padding and batching for sequences of different lengths providing a convenient external API for time-varying survival regression.

```
1 features, times, events = datasets.load_dataset('PBC', sequential = True)
2 model = DeepRecurrentSurvivalMachines(k = 3, hidden = [100], typ = 'RNN', layers = 2)
3 model.fit(features, times, events)
```
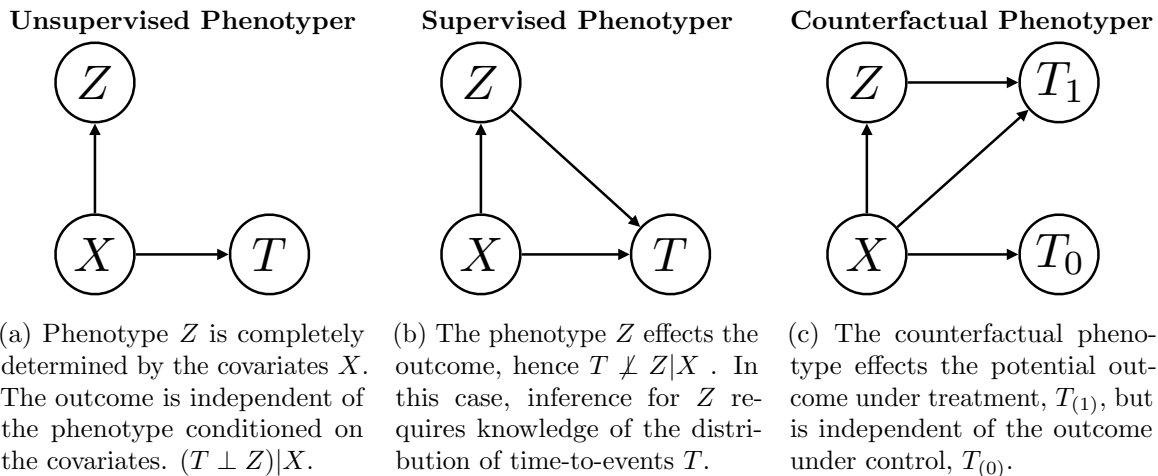
2. Phenotyping Survival Data

Unsupervised Phenotyper

(a) Phenotype Z is completely determined by the covariates X. The outcome is independent of the phenotype conditioned on the covariates. $(T \perp Z)|X$.

Supervised Phenotyper

(b) The phenotype Z effects the outcome, hence $T \not\perp Z|X$. In this case, inference for Z requires knowledge of the distribution of time-to-events T.

Counterfactual Phenotyper

(c) The counterfactual phenotype effects the potential outcome under treatment, $T_{(1)}$, but is independent of the outcome under control, $T_{(0)}$.

Figure 6: **DAG representations of the probabilitic phenotypers in auton-survival**: X represents the covariates, T the time-to-event and Z is the phenotype to be inferred.

Event incidence (survival) rates differ across groups of individuals with heterogeneous characteristics. Identifying groups of patients with similar survival rates can be used to derive insight about practices and interventions that can help improve longevity for such groups. While domain knowledge can help identify such subgroups, in practice there could be potentially complex, non-linear feature interactions that determine assignment to such groups, making identifiability difficult. In auton-survival, we refer to this group identification and survival assessment as *phenotyping*.

auton-survival offers multiple approaches to phenotyping that involve either the use of specific domain knowledge as in the case of the intersectional phenotyper, or a completely unsupervised approach that clusters based on the observed covariates. Additionally, auton-survival also offers phenotypers that explicitly involve supervision in the form of the observed outcomes and counterfactual to better inform the learnt phenotypes. Figure 6 demonstrates the probabilistic phenotypers offered by auton-survival as well as the corresponding assumption of conditional independence encoded in the models.

2.1. Intersectional Phenotyping

The IntersectionalPhenotyper class recovers groups, or phenotypes, of individuals over exhaustive combinations of user-specified categorical and numerical features. Numeric covariates are binned based on user-specified quantiles. The intersectional phenotyper is *unsupervised*, but does not have an explicit probabilistic interpretation. Figure 7 presents the intersectional phenogroups of cancer status and age group extracted on SUPPORT.
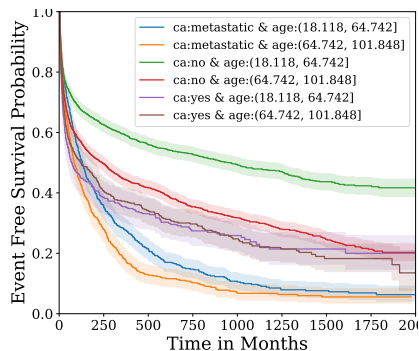

Figure 7: Intersectional Phenotypes.

```
1 from auton_survival.phenotyping import IntersectionalPhenotyper        [Demo Notebook]
2 # 'ca' is cancer status. 'age' is binned into two quantiles.
3 phenotyper = IntersectionalPhenotyper(num_vars_quantiles=(0, .5, 1.0),
4                        cat_vars=['ca'], num_vars=['age'])
5 phenotypes = phenotyper.fit_phenotype(features)
6 # Plot the phenogroup specific Kaplan-Meier survival estimate.
7 auton_survival.reporting.plot_kaplanmeier(outcomes, phenotypes)
```

2.2. Unsupervised Phenotyping

Unsupervised phenotyping identifies groups based on structured similarity in the space of features. The ClusteringPhenotyper class achieves this by first performing dimensionality reduction of the input covariates $\boldsymbol{x}$, followed by clustering. The estimated probability of an individual to belong to a latent group Z is computed as the distance to the cluster normalized by the sum of distance to other clusters. Thus,

$$\mathbb{P}(Z = k | X = \boldsymbol{x}) = \frac{\mathbf{d}(\boldsymbol{x}, \bar{\boldsymbol{x}}_k)}{\sum_j \mathbf{d}(\boldsymbol{x}, \bar{\boldsymbol{x}}_j)}. \tag{9}$$

Here $\mathbf{d}(\cdot)$ is the distance function corresponding to the underlying clustering method (for instance, '*euclidean*' for K-means and '*mahalonobis*' for a Gaussian Mixture Model.)

```
1 from auton_survival.phenotyping import ClusteringPhenotyper           [Demo Notebook]
2 # Dimensionality reduction using Principal Component Analysis (PCA) to 8 dimensions.
3 dim_red_method, = 'pca', 8
4 # We use a Gaussian Mixture Model (GMM) with 3 components and diagonal covariance.
5 clustering_method, n_clusters = 'gmm', 3
6 # Initialize the phenotyper with the above hyperparameters.
7 phenotyper = ClusteringPhenotyper(clustering_method=clustering_method,
8                                   dim_red_method=dim_red_method,
9                                   n_components=n_components,
10                                  n_clusters=n_clusters)
11 # Fit and infer the phenogroups.
12 phenotypes = phenotyper.fit_phenotype(features)
13 # Plot the phenogroup specific Kaplan-Meier survival estimate.
14 auton_survival.reporting.plot_kaplanmeier(outcomes, phenotypes)
```

2.3. Supervised Phenotyping

Supervised phenotyping seeks to explicitly identify latent groups of individuals with similar survival outcomes. Unlike unsupervised clustering, inferring supervised phenotypes requires time-to-events and the corresponding censoring indicators along with the covariates.

auton-survival provides utilities to perform supervised phenotyping as a direct consequence of training the *Deep Survival Machines* and *Deep Cox Mixtures* latent variable survival regression estimators. The inferred phenotypes can be obtained after learning these models using the predict_latent_z method. Note that these methods differ in the semantic meaning of the phenotypes they recover. DSM recovers phenotypes with similar parametric characteristics while DCM recovers phenotypes that adhere to proportional hazards.

Unsupervised Phenotyping

Supervised Phenotyping
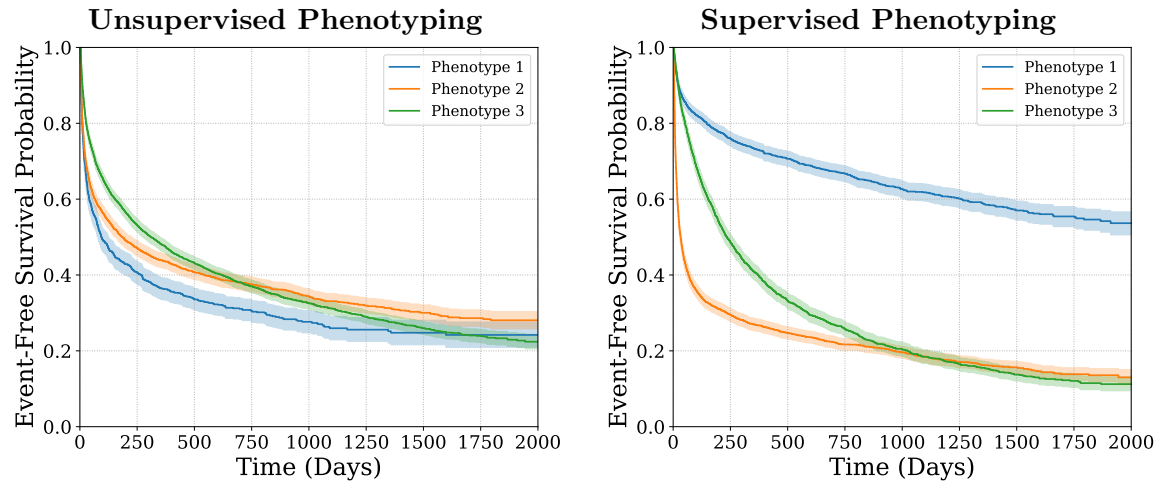
Phenotyping Purity

Phenotyper	Brier Score (BS)			Integrated BS
	1 year	**2 years**	**5 years**	**5 years**
Unsupervised	0.246 ± 0.001	0.230 ± 0.004	0.187 ± 0.015	0.218 ± 0.004
Supervised	0.234 ± 0.002	0.219 ± 0.004	0.177 ± 0.014	0.207 ± 0.004

Figure 8: Supervised phenotyping with `auton-survival` extracts phenogroups in SUPPORT with higher discriminative power as indicated by the Kaplan-Meier curves and the lower Phenotyping Purity. The intersecting survival curves suggest the phenotypers' ability to recover phenogroups that do not strictly adhere to assumptions of Proportional Hazards.

```
1 from auton_survival.models.dcm import DeepCoxMixtures          [Demo Notebook]
2 # Instantiate a DCM Model with 3 phenogroups and a single hidden layer with size 100.
3 model = DeepCoxMixtures(k = 3, layers = [100])
4 model.fit(features, outcomes.time, outcomes.event, iters = 100, learning_rate = 1e-4)
5 # Infer the latent Phenotpyes
6 latent_z_prob = model.predict_latent_z(features)
7 # Plot the phenogroup specific Kaplan-Meier survival estimate.
8 auton_survival.reporting.plot_kaplanmeier(outcomes, latent_z_prob.argmax(axis=1))
```

Quantitative Evaluation of Phenotyping

We measure a phenotyper's ability to extract subgroups, or phenogroups, with differential survival rates by fitting a Kaplan-Meier estimator within each phenogroup followed by estimating the (Integrated) Brier Score within each phenogroup. We refer to this as the *phenotyping purity*. Figure 8 demonstrates the use of phenotyping purity to compare discriminative power between unsupervised and supervised phenotyping.

```
1 from auton_survival.metrics import phenotype_purity          [Demo Notebook]
2 # Measure phenotype purity at event horizons of 1, 2 and 5 years.
3 phenotype_purity(phenotypes, outcomes, strategy='instantaneous', time=[365,730,1825])
4 # Measure phenotype purity at an event horizon of 5 years.
5 phenotype_purity(phenotypes, outcomes, strategy='integrated', time=1825)
```

2.4. Counterfactual Phenotyping

In real world clinical scenarios, individuals demonstrate heterogeneous treatment effects to an intervention. Thus, clinical decision support often requires reasoning about which groups of individuals benefit *least* (or *most*) from a certain intervention. Such identification is often referred to as *subgroup discovery* or *subgroup analysis*. While there has been extensive study in subgroup recovery with binary or continuous outcomes (Dusseldorp and Van Mechelen, 2014; Lipkovich et al., 2011; Foster et al., 2011; Nagpal et al., 2020; Wang and Rudin, 2022), censored time-to-event outcomes are less studied in the context of phenotyping.

More formally we can describe counterfactual phenotyping as estimating a function $f(\boldsymbol{x}) : \mathbb{R}^d \rightarrow \{0, 1\}$

$$f = \arg\min_{f \in \mathcal{F}} \mathbb{CATE}(\mathcal{D}|f(X) = 1) \quad \text{such that} \quad \mathbb{E}[f(X)] > \alpha. \tag{10}$$

The first term here refers to the (Conditional) Average Treatment Effect within the discovered phenotype, while the second term controls the size of the phenotype.

Virtual Twins Survival Regression: auton-survival includes a Virtual Twins model (Foster et al., 2011) involving first estimating the potential outcomes under treatment and control using a counterfactual Deep Cox Proportional Hazards model, followed by regressing the difference in the estimated counterfactual Restricted Mean Survival Times using a Random Forest regressor.

```
1 from auton_survival.phenotyping import SurvivalVirtualTwins
2 # Instantiate the Survival Virtual Twins
3 model = SurvivalVirtualTwins(horizon=365)
4 model.fit(features, outcomes.time, outcomes.event, interventions)
5 # Infer the estimated counterfactual phenotype probability.
6 phi_probs = model.predict_latent_phi(features)
```

Cox Mixture Latent Variable Model: auton-survival also includes an implementation of the Cox Mixtures with Heterogenous Effects (CMHE) (Nagpal et al., 2022) model for counterfactual phenotyping. The CMHE model involves modelling the individual hazard rate under an intervention ($A = \boldsymbol{a}$) as

$$\underbrace{\boldsymbol{\lambda}(t|X = \boldsymbol{x}, Z = k, \boldsymbol{\phi} = m, A = \boldsymbol{a})}_{\text{Conditional Hazard Rate}} = \overbrace{\boldsymbol{\lambda}_k(t)}^{\text{Base Survival Rate}} \underbrace{\exp\left(h_{\boldsymbol{\theta}}^k(\boldsymbol{x})\right)}_{\text{Effect of Confounders}} \overbrace{\exp(\omega_m)^{\boldsymbol{a}}}^{\text{Treatment Effect}}, \tag{11}$$

where ($Z = k$) is the base survival rate, ($\boldsymbol{\phi} = m$) is the latent subgroup memberships that mediates the treatment effect.

```
1 from auton_survival.models import cmhe                          [Demo Notebook]
2 # Instantiate the Cox Mixture with Heterogenous Effects model
3 model = cmhe.DeepCoxMixturesHeterogenousEffects(k=1, g=2, layers=[100])
4 model.fit(features, outcomes.time, outcomes.event, interventions,
5          iters=iters, learning_rate=learning_rate, batch_size=batch_size)
6 # Infer the estimated counterfactual phenotypes.
7 latent_phi_probs = model.predict_latent_phi(features)
```

3. Evaluation

3.1. Censoring-Adjusted Evaluation Metrics

The survival model performance can be evaluated using the following metrics, among others. auton-survival offers utilities to compute these metrics along with bootstrapped confidence intervals with a convenient API.

Brier Score (BS): The Brier Score involves computing the Mean Squared Error of the estimated survival probabilities, $\widehat{\mathbb{P}}(T > t|X = \boldsymbol{x}) = f(\boldsymbol{x}, t)$ at a specified time horizon, t. As a proper scoring rule, Brier Score gives a sense of both discrimination and calibration.

Under the assumption that the censoring distribution is independent of the time-to-event, we can obtain an unbiased, censoring adjusted estimate of the Brier Score $\widehat{\mathrm{BS}}_{\mathrm{IPCW}}(t)$ using inverse probability of censoring weights (IPCW) from a Kaplan-Meier estimator of the censoring distribution, $\hat{G}(.)$ as proposed in Graf et al. (1999); Gerds and Schumacher (2006).

$$\mathrm{BS}(t) = \mathbb{E}_{\mathcal{D}}\left[\left(\mathbf{1}\{T_i > t\} - \widehat{\mathbb{P}}(T > t|X = \boldsymbol{x})\right)^2\right]$$

$$\widehat{\mathrm{BS}}_{\mathrm{IPCW}}(t) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{f(\boldsymbol{x}_i, t)^2 \mathbf{1}\{T \le t, \delta_i = 1\}}{\hat{G}_i(T_i)} + \frac{(1 - f(\boldsymbol{x}_i, t))^2 \mathbf{1}\{T > t\}}{\hat{G}_i(t)}\right].$$

Area under ROC Curve (AUC): ROC Curves are extensively used in classification tasks where the true positive rate (TPR) is plotted against the false positive rate (1-true negative rate, or TNR) to measure the in model discriminative power over all output thresholds.

To enable the use of ROC curves to assess the performance of survival models subject to censoring, we employ the technique proposed by Uno et al. (2007); Hung and Chiang (2010), which treats the TPR as time-dependent on a specified horizon, t and adjusts survival probabilities, $f(\boldsymbol{x}, t)$, using the IPCW from a Kaplan-Meier estimator of the censoring distribution, $\hat{G}(t)$. Estimating the TNR requires observing outcomes for each individual, only uncensored instances are used. Refer to Kamarudin et al. (2017) for details on computing ROC curves in the presence of censoring.

$$\widehat{\mathrm{TPR}}(c, t) = \frac{\sum_{i=1}^{n}\frac{\delta_i}{\hat{G}(T_i)} \cdot \mathbf{1}\{f(\boldsymbol{x}_i, t) > c, T_i \le t\}}{\sum_{i=1}^{n}\frac{\delta_i}{\hat{G}(T_i)} \cdot \mathbf{1}\{T_i < t\}\cdot}; \quad \widehat{\mathrm{TNR}}(c, t) = \frac{\sum_{i=1}^{n}\mathbf{1}\{f(\boldsymbol{x}_i, t) \le c, T_i > t\}}{\sum_{i=1}^{n}\mathbf{1}\{T_i > t\}}$$

Time Dependent Concordance Index (C^{td}): Concordance Index compares risks across all pairs of individuals within a fixed time horizon, t, to estimate ability to appropriately rank instances relative to each other in terms of their risks, $f(\boldsymbol{x}, t)$.

$$C^{td}(t) = \mathbb{P}\big(f(\boldsymbol{x}_i, t) \le f(\boldsymbol{x}_j, t)|\delta_i = 1, T_i < T_j, T_i \le t\big)$$

We employ the censoring adjusted estimator for C^{td} that exploits IPCW estimates from a Kaplan-Meier estimate of the censoring distribution. Further details can be found in Uno et al. (2011) and Gerds et al. (2013).

```
1 from auton_survival.metrics import survival_regression_metric
2 # Infer event-free survival probability from model
3 predictions = model.predict_survival(features, times)
4 # Compute Brier Score, Integrated Brier Score
5 # Area Under ROC Curve and Time Dependent Concordance Index
6 metrics = ['brs', 'ibs', 'auc', 'ctd']
7 for metric in metrics
8     score = survival_regression_metric(metric, outcomes_train, outcomes_test,
                                         predictions_test, times=times)
```

3.2. Comparing Treatment Arms

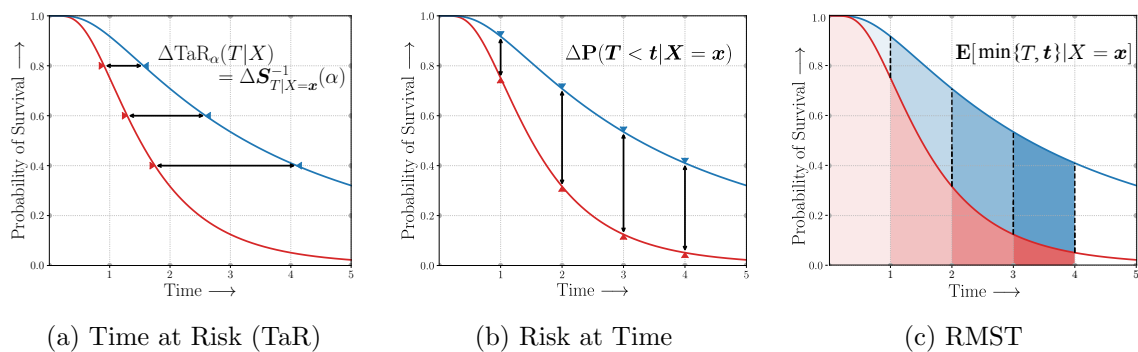(a) Time at Risk (TaR) (b) Risk at Time (c) RMST

Figure 9: Treatment effects measured in terms of the difference in metrics computed for treatment and control groups including (a) the time at a specified level of risk (b) risk at a certain time and (c) the expected survival time over a truncated time horizon.

Hazard Ratio

Assuming the proportional hazards assumptions holds, the treatment effect can be modelled as the ratio of hazard rates between the treatment and control arms. This corresponds to fitting a univariate Cox Proportional Hazards model and is arguably the most popular approach in estimating treatment effects for censored time-to-events.

Time at Risk (TaR)

Time at Risk (TaR) (Figure 9a) measures time-to-event at a specified level of risk. Computing TaR may be of interest when maximum permissible outcome risk is predefined and the corresponding time-to-event is the deciding factor for assessing treatment benefit at that risk level.

$$\text{TaR}_\alpha(T|X) = \sup_t \left\{ t \in \mathbb{R}^+ : \boldsymbol{S}_{T|X}(t) < \alpha \right\} = \boldsymbol{S}_{T|X}^{-1}(\alpha) \tag{12}$$

Risk at Time

Risk at Time (Figure 9b) measures risk at a specified time horizon. This metric may be of interest when treating survival as a binary outcome over a specific time horizon.

$$\text{Risk-at-Time}(t|X = \boldsymbol{x}) = \mathbb{P}(T < t|X = \boldsymbol{x}) = 1 - \boldsymbol{S}(t|X = \boldsymbol{x}) \tag{13}$$

Restricted Mean Survival Time (RMST)

The RMST (Figure 9c) is the expected (or mean) time-to-event conditioned on a specified time horizon. Mathematically, the RMST is $\mathbb{E}[\min\{T, \boldsymbol{t}\}|X = \boldsymbol{x}]$ and translates into the area under the survival curve till time $\boldsymbol{t}$.

$$\text{RMST}_{\boldsymbol{t}}(T|X = \boldsymbol{x}) = \mathbb{E}[\min\{T, \boldsymbol{t}\}|X = \boldsymbol{x}] \tag{14}$$

$$= \int_0^t \boldsymbol{S}(\boldsymbol{t}|X = \boldsymbol{x})\mathrm{d}\boldsymbol{t}.$$

RMST might be preferred over Hazard Ratio in comparing treatment arms where there is reason to believe that the proportional hazards assumption is violated.

```
1  from auton_survival.metrics import treatment_effect
2  # Compute the difference in RMST between treatment and control groups
3  effect = treatment_effect(metric='restricted_mean', outcomes=outcomes
4                            treatment_indicator=treatment_indicator,
5                            weights=None, horizon=120, n_bootstrap=500)
```

3.3. Propensity Adjusted Treatment Effects

Differences in the values of metrics specified in Section 3.2 across treatment arms is often averaged over the population to estimate net benefit or the average treatment ($\mathbb{ATE}$),

$$\mathbb{ATE}(\mathcal{D}, f) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}} \left[\mathbb{E}[f_1(x) - f_0(x)|X = \boldsymbol{x}]\right]$$

Directly computing the metrics in above in an observational settings would result in mis-estimations of treatment effects as it would not adjust for potential confounders that influence both treatment assignment and the outcome. auton-survival supports the computation of propensity-adjusted treatment effects in terms of the above metrics through bootstrap resampling the dataset, $\mathcal{D}$ with replacement. Here the resampling weights are obtained using inverse propensity of treatment weighting (IPTW). The bootstrapped treatment effect thus converges to the Thompson-Horvitz estimate of the Treatment Effect.

$$\mathbb{ATE}(\mathcal{D}^*, f) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}^*} \left[\mathbb{E}[f_1(x) - f_0(x)|X = \boldsymbol{x}]\right]; \quad \mathcal{D}^* \sim \frac{1}{\widehat{\mathbb{P}}(A|X)} \cdot \mathbb{P}^*(\mathcal{D}) \tag{15}$$

The auton-survival function treatment_effect function supports sample weight inputs in the form of treatment propensity scores, such as obtained from scaled classification model scores that estimate probability of treatment.

```
1  from auton_survival.metrics import treatment_effect
2  from sklearn.linear_model import LogisticRegression
3  # Train a classification model to compute "treatment" propensity scores
4  model = LogisticRegression(penalty='l2').fit(features, treatment_indicator)
5  treatment_propensity = model.predict_proba(features)[:, 1]
6  # Compute the treatment effect after adjusting for treatment propensity
7  adjusted_effect = treatment_effect(metric='hazard_ratio', outcomes=outcomes
8                            treatment_indicator=treatment_indicator,
9                            weights=treatment_propensity, n_bootstrap=500)
```

4. Case Study: Regional disparities in Breast Cancer Incidence

Sample Size and RMST by Region

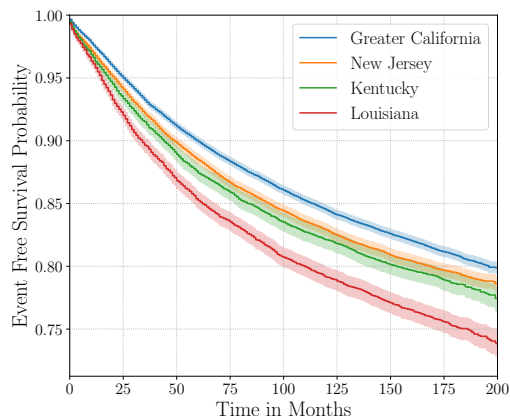| Region	Size	RMST
Greater California	47,757	108.9 ± 0.261
New Jersey	27,665	107.3 ± 0.352
Greater Georgia	14,212	106.8 ± 0.559
Louisiana	11,837	104.1 ± 0.598
Kentucky	11,581	106.4 ± 0.610
All Regions	113,052	107.5 ± 0.189

Figure 10: Breast cancer survival from the cohort subset of the SEER study. Notice discrepancies in the Kaplan-Meier estimated survival rates when stratified by the geographic region. Similarly, regional discrepancies in expected time-to-event are highlighted by mean (95% CI) RMST computed for bootstrap samples conditioned on a 10-year horizon.

The SEER registry established by the National Cancer Institute consists of patient demographic and tumour morphology data and long term incidence information of approximately one-third of the U.S. population (Ries et al., 2008). This study is a retrospective analysis of the SEER[2] data. For this study, we consider a subset of 113,052 patients from Greater California, Kentucky, Louisiana, New Jersey and Georgia diagnosed with breast cancer between a four year period from January 2000 to December 2003[3]. We consider all-cause mortality as the outcome of interest with a maximum follow up period of 18 years from January, 2000. Along with outcomes we also consider additional variables including age, race, tumour morphology and treatments administered. A complete list of the variables considered in this study can be found in Appendix B.

4.1. The Effect of Geographical Region on Breast Cancer Mortality

It is evident that there are discrepancies between the observed mortality rates stratified by the geographic region (Figure 10). A natural question arises as to whether this discrepancy can be attributed to geographic region or other socio-economic or physiological confounding factors that effect both belonging to these regions and the outcome.

We attempt to provide insight into this question by assessing the effect size via the Potential Outcomes framework by considering the region as an intervention. We first estimate the counterfactual survival rates across treatment arms to evaluate the effect of treatment on event-free survival rates. We further verify our findings by comparing treatment effects before and after adjusting for treatment propensity by inverse propensity weighting.

---

2. Surveillance, Epidemiology and End Results - National Cancer Institute
3. We restrict our study to this cohort subset as it had a consistent coding pattern vis-a-vis tumour morhology and therapy characteristics in the original SEER database

COUNTERFACTUAL SURVIVAL ESTIMATION

To adjust estimates of survival with counterfactual estimation, we train two separate Deep Cox models as described in Section 1.3 on data from Greater California and Louisiana as counterfactual regressors. The fitted regressors are then applied to estimate the survival curves for each instance, which are then averaged over treatment groups to compute the domain-specific survival rate.

Figure 11 presents the counterfactual survival rates compared with the survival rates obtained from a Kaplan-Meier estimator. The Kaplan-Meier estimator does not adjust for confounding and so overestimates treatment effect as evidenced from the extent that survival rates differ between regions. Counterfactual regression adjusts for confounding factors and predicts more similar survival rates between regions.
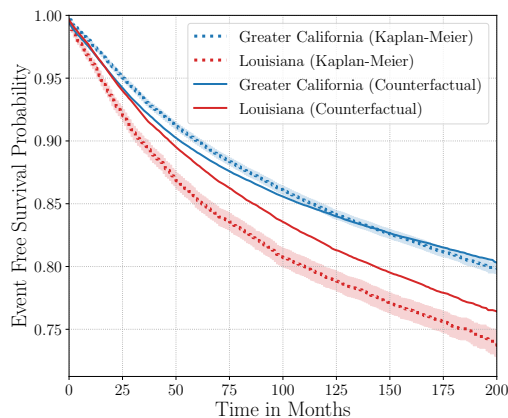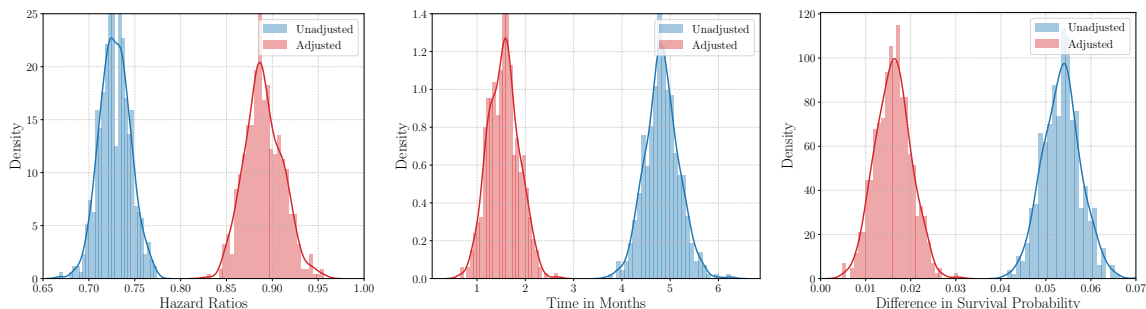

Figure 11: The Kaplan-Meier estimator overestimates effect of region on mortality compared with the counterfactual regression model, which reduces discrepancy in survival rates between regions.

PROPENSITY ADJUSTED TREATMENT EFFECTS

| (a) Hazard Ratios | (b) RMST | (c) Risk Difference |

Treatment Effect	10-year Event Horizon		
	Hazard Ratio	RMST	Risk Difference
Unadjusted	0.728 ± 0.032	4.829 ± 0.684	0.054 ± 0.008
Adjusted	0.890 ± 0.040	1.549 ± 0.618	0.016 ± 0.008

Figure 12: Estimated probability densities of the bootstrapped treatment effects to assess discrepancies in survival across regions. When bootstrapping is performed with weighted sampling based on region propensity to adjust for confounders, the discrepancies in survival between Greater California and Louisiana are much less pronounced. Bootstrap mean (standard deviation) results shown in the table summarize the observed mitigated difference.

A alternative method to assess effect size of the geographical region on survival involves comparing differences after adjusting for treatment propensity. We trained a Logistic Regression with an ℓ_2 penalty by regression the Geographical Region on the set of confounding variables. The propensity scores were then employed as sampling weights for the treatment effects in terms of hazard ratios, restricted mean survival time (RMST), and risk difference as in Figure 12. Adjusting for region propensity noticeably mitigates differences in treatment effects, indicating that mortality due to breast cancer is likely explained by confounding socio-economic and physiological factors rather than solely geographic region.

```
1  from auton_survival.metrics import treatment_effect
2  from sklearn.linear_model import LogisticRegression
3  # Create "treatment" labels
4  domain = regions=='Louisiana'
5  # Train a logistic regression model to compute "domain" propensity scores
6  model = LogisticRegression(penalty='l2').fit(features, domain)
7  propensity = model.predict_proba(features)[:, 1]
8  # Compute the treatment effect before and after adjusting for treatment propensity
9  effect = treatment_effect(metric='hazard_ratio', outcomes, domain,
10                            n_bootstrap=500)
11 adjusted_effect = treatment_effect(metric='hazard_ratio', outcomes, domain,
                                      weights=propensity, n_bootstrap=500)
```

4.2. Domain Adaptation

Area under ROC Curve

Louisiana → Greater California				Greater California → Louisiana			
Horizon	1-Year	2-Year	5-Year	Horizon	1-Year	2-Year	5-Year
Unadjusted	0.8398	0.8177	0.8010	Unadjusted	0.8598	0.8343	0.8163
Adjusted	0.8449	0.8220	0.8063	Adjusted	0.8603	0.8348	0.8182

Brier Score

Louisiana → Greater California				Greater California → Louisiana			
Horizon	1-Year	2-Year	5-Year	Horizon	1-Year	2-Year	5-Year
Unadjusted	0.0675	0.0967	0.1154	Unadjusted	0.0858	0.1119	0.1312
Adjusted	0.0673	0.0963	0.1148	Adjusted	0.0856	0.1118	0.1307

Table 1: Domain Adaptation with importance weighting improves both calibration and discriminative performance when applying models trained on one domain to the other.

Consider a scenario when training data is available from Greater California (Lousiana) and is to be used to train models to estimate risk for patients in Lousiana (Greater California). Discrepancies in the data distributions would naturally also translate into poorer generalization when using these models for individual level survival regression.

It would thus help adjusting for these distributional differences using IWERM as introduced Section 1.2. We train a logistic regression with an ℓ_2 penalty to estimate the

probability of an individual belonging to one of the two regions. The estimated probabilities are then used as importance weights to resample th individuals when training a Cox Model.

Table 1 present the performance of a linear Cox model in terms of discriminative performance (Area under ROC curve) and well as calibration (Brier Score) at predicting all cause mortality within 5, 10 and 15 years of entry into the study. Notice that the importance weighted Cox model has better discriminative performance and calibration as compared to the the unadjusted model.

```python
from auton_survival.estimators import SurvivalModel
from sklearn.linear_model import SGDClassifier
regions=['Greater California', 'Louisiana']
for source in regions:
    for target in regions:
        # Train a Logistic Regression to estimate treatment propensity scores
        domain_model = SGDClassifier(loss='log', penalty='l2', alpha=.1)
        domain_model.fit(features, regions==target)
        probs = domain_model.predict_proba(features[regions==source])[:, 1]
        # Use propensity scores to compute sample weights
        weights = ((probs)/(1-probs))**0.2
        # Fit DCPH model with weighted resampling
        model = SurvivalModel(model='dcph', layers=[100])
        model.fit(features[regions==source], outcomes[regions==source],
                  weights, resample_size=10.0)
```

Conclusion

We presented `auton-survival`, an open source `Python` package encapsulating multiple pipelines to analyze censored time-to-event data ubiquitous in healthcare. Through the use of multiple code examples, notebooks, and illustrations we demonstrate the efficacy of `auton-survival` to analyze complex healthcare data and answer clinical and epidemiological questions including real-world case studies.

We acknowledge `auton-survival` as one step towards promoting reproducible machine learning research for healthcare. We believe that the broader use of software packages such as ours will help accelerate and systematize impactful machine learning research for healthcare. We expect that ongoing and future collaborations with the machine learning and healthcare communities will allow us to further enhance the collection of open-source survival regression methodologies that can support reproducible analysis of censored time-to-event data.

Acknowledgements

References

Paidamoyo Chapfuwa, Chenyang Tao, Chunyuan Li, Courtney Page, Benjamin Goldstein, Lawrence Carin Duke, and Ricardo Henao. Adversarial time-to-event modeling. In *International Conference on Machine Learning*, pages 735–744. PMLR, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Alfred F Connors, Neal V Dawson, Norman A Desbiens, William J Fulkerson, Lee Goldman, William A Knaus, Joanne Lynn, Robert K Oye, Marilyn Bergner, Anne Damiano, et al. A controlled trial to improve care for seriously iii hospitalized patients: The study to understand prognoses and preferences for outcomes and risks of treatments (support). *Jama*, 274(20):1591–1598, 1995.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972.

Cameron Davidson-Pilon. lifelines: survival analysis in python. *Journal of Open Source Software*, 4(40):1317, 2019.

Elise Dusseldorp and Iven Van Mechelen. Qualitative interaction trees: a tool to identify qualitative treatment–subgroup interactions. *Statistics in medicine*, 33(2):219–237, 2014.

David Faraggi and Richard Simon. A neural network model for survival data. *Statistics in medicine*, 14(1):73–82, 1995.

Jared C Foster, Jeremy MG Taylor, and Stephen J Ruberg. Subgroup identification from randomized clinical trial data. *Statistics in medicine*, 30(24):2867–2880, 2011.

Thomas A Gerds and Martin Schumacher. Consistent estimation of the expected brier score in general survival models with right-censored event times. *Biometrical Journal*, 48(6): 1029–1040, 2006.

Thomas A Gerds, Michael W Kattan, Martin Schumacher, and Changhong Yu. Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring. *Statistics in Medicine*, 32(13):2173–2184, 2013.

Erika Graf, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18(17-18):2529–2545, 1999.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Hung Hung and Chin-tsang Chiang. Optimal composite markers for time-dependent receiver operating characteristic curves with censored survival data. *Scandinavian journal of statistics*, 37(4):664–679, 2010.

H. Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3), 2008.

Adina Najwa Kamarudin, Trevor Cox, and Ruwanthi Kolamunnage-Dona. Time-dependent roc curve analysis in medical research: current methods and applications. *BMC medical research methodology*, 17(1):53, 2017.

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): 1–12, 2018.

W. A. Knaus, Harrell F. E., Lynn J, and et al. The support prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, 122: 191–203, 1995.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and cox regression. *arXiv preprint arXiv:1907.00825*, 2019.

Changhee Lee, William Zame, Jinsung Yoon, and Mihaela Van Der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hyun Gi Lee, Evan Sholle, Ashley Beecy, Subhi Al'Aref, and Yifan Peng. Leveraging deep representations of radiology reports in survival analysis for predicting heart failure patient mortality. *arXiv preprint arXiv:2105.01009*, 2021.

Ilya Lipkovich, Alex Dmitrienko, Jonathan Denne, and Gregory Enas. Subgroup identification based on differential effect search—a recursive partitioning method for establishing response to treatment in patient subpopulations. *Statistics in medicine*, 30(21):2601–2621, 2011.

Chirag Nagpal, Dennis Wei, Bhanukiran Vinzamuri, Monica Shekhar, Sara E Berger, Subhro Das, and Kush R Varshney. Interpretable subgroup discovery in treatment effect estimation with application to opioid prescribing guidelines. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, pages 19–29, 2020.

Chirag Nagpal, Vincent Jeanselme, and Artur Dubrawski. Deep parametric time-to-event regression with time-varying covariates. In *Survival Prediction-Algorithms, Challenges and Applications*, pages 184–193. PMLR, 2021a.

Chirag Nagpal, Xinyu Li, and Artur Dubrawski. Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE Journal of Biomedical and Health Informatics*, 25(8):3163–3175, 2021b.

Chirag Nagpal, Steve Yadlowsky, Negar Rostamzadeh, and Katherine Heller. Deep cox mixtures for survival regression. In *Machine Learning for Healthcare Conference*, pages 674–708. PMLR, 2021c.

Chirag Nagpal, Mononito Goswami, Keith Dufendach, and Artur Dubrawski. Counterfactual phenotyping with censored time-to-events. *arXiv preprint arXiv:2202.11089*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12: 2825–2830, 2011.

Sebastian Pölsterl. scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *J. Mach. Learn. Res.*, 21(212):1–6, 2020.

Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. Deep survival analysis. In *Machine Learning for Healthcare Conference*, pages 101–114. PMLR, 2016.

LAG Ries, D Melbert, M Krapcho, DG Stinchcomb, N Howlader, MJ Horner, A Mariotto, BA Miller, EJ Feuer, SF Altekruse, et al. Seer cancer statistics review, 1975–2005. *Bethesda, MD: National Cancer Institute*, 2999, 2008.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

Hajime Uno, Tianxi Cai, Lu Tian, and Lee-Jen Wei. Evaluating prediction rules for t-year survivors with censored regression models. *Journal of the American Statistical Association*, 102(478):527–537, 2007.

Hajime Uno, Tianxi Cai, Michael J Pencina, Ralph B D'Agostino, and LJ 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.

Tong Wang and Cynthia Rudin. Causal rule sets for identifying subgroups with enhanced treatment effects. *INFORMS Journal on Computing*, 2022.

Chun-Nam Yu, Russell Greiner, Hsiu-Chin Lin, and Vickie Baracos. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. *Advances in neural information processing systems*, 24, 2011.

Appendix A. Comparison to Other Packages

In the code snippets below we will compare the API of `auton-survival` with the popular alternative, `pycox` to train a Deep Cox PH model on the SUPPORT dataset.

```python
from auton_survival import datasets, preprocessing
# Load the SUPPORT Dataset
outcomes, features = datasets.load_dataset("SUPPORT")
# Preprocess (Impute and Scale) the features
features = preprocessing.Preprocessor().fit_transform(features)
```

Training a Deep Cox Proportional Hazards model with `pycox`

```python
from torchtuples.tt.practical import MLPVanilla
from pycox.models import CoxPH
net = MLPVanilla(features.shape[1], [100], 1, output_bias=False) # Instantiate an MLP
model = CoxPH(net, tt.optim.Adam) # Instantiate a pycox model.
# Fit the pycox model.
model.fit(features, (outcomes.time, outcomes.event), batch_size=256, epochs=10)
```

Training a Deep Cox Proportional Hazards model with `auton-survival`.

```python
from auton_survival import models
# Train a Deep Cox Proportional Hazards (DCPH) model
model = models.cph.DeepCoxPH(layers=[100])
model.fit(features, outcomes.time, outcomes.event, batch_size=256,)
# Predict risk at specific time horizons.
predictions = model.predict_risk(features, t=[8, 12, 16])
```

Utility / Package	lifelines	pycox	scikit-survival	auton-survival
Deep Survival Models	✗	✓	✗	✓
Time-Varying Survival Analysis	✓	✗	✗	✓
Counterfactual Estimation	✗	✗	✗	✓
Subgroup Identification (Phenotyping)	✗	✗	✗	✓
Treatment Effect Estimation	✓	✗	✗	◯
Regression Metrics	✓	✓	✓	◯
Cross-Validation	✗	✗	✗	✓
Preprocessing	✗	✓	✓	✓
Documentation and Examples	✓	✓	✓	✓

Table 2: A comparison of `auton-survival` to other open-source `python` packages for survival analysis. ◯ indicates that the utility depends on one of other survival analysis packages.

Appendix B. Additional details on the SEER case study

Name	Description
AGE	Age of participant
SEX	Sex of participant
RACE1V	Race/Ethnicity of participant
PRIMSITE	Site in which the primary tumor originated.
LATERAL	Side of a paired organ or body on which the tumor originated.
HISTO3V	Histologic Type ICD-O-3.
BEHO3V	Behavior Code ICD-O-3.
DX_CONF	Diagnostic Confirmation.
SURGPRIF	Surgery of Primary Site.
SURGSITF	The surgical removal of distant tissue beyond the primary site.
NUMNODES	Number of Examined Nodes
NO_SURG	Reason for no surgery.
SURGSITE	The removal of distant tissue beyond the primary site.
EOD10_SZ	Tumor size.
EOD10_EX	Tumor extension.
EOD10_ND	The highest specific lymph node chain that is involved by the tumor.
EOD10_PN	Regional nodes positive.
EOD10_NE	regional nodes examined.
MALIGCOUNT	Total number of In Situ/malignant tumors for patient.
BENBORDCOUNT	Total number of benign/borderline tumors.

Table 3: List of confounding features used for experiments involving the the SEER dataset.