

# Package ‘survdnn’

April 29, 2026

**Title** Deep Neural Networks for Survival Analysis with R 'torch'

**Version** 0.7.6

**Description** Provides deep learning models for right-censored survival data using the 'torch' backend. Supports multiple loss functions, including Cox partial likelihood, L2-penalized Cox, time-dependent Cox, and accelerated failure time (AFT) loss. Offers a formula-based interface, built-in support for cross-validation, hyperparameter tuning, survival curve plotting, and evaluation metrics such as the C-index, Brier score, and integrated Brier score. For methodological details, see Kvamme et al. (2019) <<https://www.jmlr.org/papers/v20/18-424.html>>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** torch, survival, stats, utils, tibble, dplyr, purrr, tidyr, ggplot2, methods, rsample, cli, glue

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

**BugReports** <https://github.com/ielbadisy/survdnn/issues>

**URL** <https://github.com/ielbadisy/survdnn>

**NeedsCompilation** no

**Author** Imad EL BADISY [aut, cre]

**Maintainer** Imad EL BADISY <elbadisyimad@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-04-29 06:40:13 UTC

## Contents

brier . . . . .	2
callback_early_stopping . . . . .	3
cindex_survmat . . . . .	4
cv_survdnn . . . . .	4
evaluate_survdnn . . . . .	6
gridsearch_survdnn . . . . .	7
ibs_survmat . . . . .	8
plot.survdnn . . . . .	9
plot_loss . . . . .	10
predict.survdnn . . . . .	11
print.survdnn . . . . .	12
summarize_cv_survdnn . . . . .	12
summarize_tune_survdnn . . . . .	13
summary.survdnn . . . . .	14
survdnn . . . . .	15
tune_survdnn . . . . .	17
<b>Index</b>	<b>19</b>

---

brier	<i>Brier Score for Right-Censored Survival Data at a Fixed Time</i>
-------	---

---

### Description

Computes the Brier score at a fixed time point using inverse probability of censoring weights (IPCW).

### Usage

```
brier(object, pre_sp, t_star)
```

### Arguments

object	A ‘Surv’ object with observed time and status.
pre_sp	A numeric vector of predicted survival probabilities at ‘t_star’.
t_star	The evaluation time point.

### Value

A single numeric value representing the Brier score a a specific time point.

## Examples

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  veteran <- survival::veteran
  mod <- survdnn(survival::Surv(time, status) ~
    age + karno + celltype, data = veteran, epochs = 50, verbose = FALSE)
  pred <- predict(mod, newdata = veteran, type = "survival", times = c(30, 90, 180))
  y <- model.response(model.frame(mod$formula, veteran))
  survdnn::brier(y, pre_sp = pred[["t=90"]], t_star = 90)
}
```

---

callback\_early\_stopping

*Early stopping callback for survdnn*

---

## Description

Simple early stopping on a monitored scalar, typically the training loss.

## Usage

```
callback_early_stopping(
  patience = 10L,
  min_delta = 0,
  mode = c("min", "max"),
  verbose = FALSE
)
```

## Arguments

patience	Integer. Number of epochs with no improvement before stopping.
min_delta	Minimum change to qualify as an improvement (default: 0).
mode	Character. "min" (for losses) or "max" (for metrics to maximize).
verbose	Logical. Whether to print a message when early stopping is triggered.

## Value

A function of the form ‘function(epoch, current)’ that returns TRUE if training should stop, FALSE otherwise.

---

cindex\_survmat                      *Concordance Index from a Survival Probability Matrix*

---

### Description

Computes the time-dependent concordance index (C-index) from a predicted survival matrix at a fixed time point. The risk is computed as  $1 - S(t\_star)$ .

### Usage

```
cindex_survmat(object, predicted, t_star = NULL)
```

### Arguments

object	A 'Surv' object representing the observed survival data.
predicted	A data frame or matrix of predicted survival probabilities. Each column corresponds to a time point (e.g., 't=90', 't=180').
t_star	A numeric time point corresponding to one of the columns in 'predicted'. If 'NULL', the last column is used.

### Value

A single numeric value representing the C-index.

### Examples

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  veteran <- survival::veteran
  mod <- survdnn(survival::Surv(time, status) ~
    age + karno + celltype, data = veteran, epochs = 50, verbose = FALSE)
  pred <- predict(mod, newdata = veteran, type = "survival", times = c(30, 90, 180))
  y <- model.response(model.frame(mod$formula, veteran))
  cindex_survmat(y, pred, t_star = 180)
}
```

---

cv\_survdnn                      *K-Fold Cross-Validation for survdnn Models*

---

### Description

Performs cross-validation for a 'survdnn' model using the specified evaluation metrics.

**Usage**

```

cv_survdnn(
  formula,
  data,
  times,
  metrics = c("cindex", "ibs"),
  folds = 5,
  .seed = NULL,
  .device = c("auto", "cpu", "cuda"),
  .threads = NULL,
  na_action = c("omit", "fail"),
  verbose = TRUE,
  ...
)

```

**Arguments**

formula	A survival formula, e.g., 'Surv(time, status) ~ x1 + x2'.
data	A data frame.
times	A numeric vector of evaluation time points.
metrics	A character vector: any of "cindex", "brier", "ibs".
folds	Integer. Number of folds to use.
.seed	Optional. Set random seed for reproducibility.
.device	Character string indicating the computation device used when fitting the models in each fold. One of "auto", "cpu", or "cuda". "auto" uses CUDA if available, otherwise falls back to CPU.
.threads	Optional positive integer. If provided, sets Torch CPU thread count before each fold fit via 'torch::torch_set_num_threads()'.
na_action	Character. How to handle missing values within each fold: "omit" drops incomplete rows; "fail" errors if any NA is present.
verbose	Logical; whether to print cross-validation progress and propagate verbose messages to fitting/evaluation in each fold (default: TRUE).
...	Additional arguments passed to [survdnn()].

**Value**

A tibble containing metric values per fold and (optionally) per time point.

**Examples**

```

if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  veteran <- survival::veteran
  cv_survdnn(
    survival::Surv(time, status) ~ age + karno + celltype,
    data = veteran,
    times = c(30, 90, 180),
  )
}

```

```

  metrics = "ibs",
  folds = 3,
  .seed = 42,
  hidden = c(16, 8),
  epochs = 5
)
}

```

---

 evaluate\_survdnn

*Evaluate a survdnn Model Using Survival Metrics*


---

### Description

Computes evaluation metrics for a fitted ‘survdnn’ model at one or more time points. Supported metrics include the concordance index (“cindex”), Brier score (“brier”), and integrated Brier score (“ibs”).

### Usage

```

evaluate_survdnn(
  model,
  metrics = c("cindex", "brier", "ibs"),
  times,
  newdata = NULL,
  na_action = c("omit", "fail"),
  verbose = FALSE
)

```

### Arguments

model	A fitted ‘survdnn’ model object.
metrics	A character vector of metric names: “cindex”, “brier”, “ibs”.
times	A numeric vector of evaluation time points.
newdata	Optional. A data frame on which to evaluate the model. Defaults to training data.
na_action	Character. How to handle missing values in evaluation data: “omit” drops incomplete rows, “fail” errors if any NA is present.
verbose	Logical. If TRUE and ‘na_action=“omit”’, prints a message when rows are removed.

### Value

A tibble with evaluation results, containing at least ‘metric’, ‘value’, and possibly ‘time’.

---

**gridsearch\_survdmn** *Grid Search for survdmn Hyperparameters*

---

**Description**

Performs grid search over user-specified hyperparameters and evaluates performance on a validation set.

**Usage**

```
gridsearch_survdmn(  
  formula,  
  train,  
  valid,  
  times,  
  metrics = c("cindex", "ibs"),  
  param_grid,  
  .seed = 42,  
  .device = c("auto", "cpu", "cuda")  
)
```

**Arguments**

formula	A survival formula (e.g., 'Surv(time, status) ~ .')
train	Training dataset
valid	Validation dataset
times	Evaluation time points (numeric vector)
metrics	Evaluation metrics (character vector): any of "cindex" and "ibs".
param_grid	A named list of hyperparameters to search over. Currently supported entries are hidden, lr, activation, epochs, and loss.
.seed	Optional random seed for reproducibility
.device	Character string indicating the computation device used when fitting all models in the grid search. One of "auto", "cpu", or "cuda". This is a runtime setting and is not part of the hyperparameter grid.

**Value**

A tibble with configurations and their validation metrics

**Examples**

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {  
  set.seed(123)  
  
  # Simulate small dataset  
  n <- 60
```

```

x1 <- rnorm(n); x2 <- rbinom(n, 1, 0.5)
time <- rexp(n, rate = 0.1)
status <- rbinom(n, 1, 0.7)
df <- data.frame(time, status, x1, x2)

# Split into training and validation
idx <- sample(seq_len(n), 0.7 * n)
train <- df[idx, ]
valid <- df[-idx, ]

# Define formula and param grid
formula <- survival::Surv(time, status) ~ x1 + x2
param_grid <- list(
  hidden    = list(c(4)),
  lr        = c(1e-3),
  activation = c("relu"),
  epochs    = c(1),
  loss      = c("cox")
)

# Run grid search
results <- gridsearch_survdnn(
  formula = formula,
  train   = train,
  valid   = valid,
  times   = c(10, 20),
  metrics = c("cindex", "ibs"),
  param_grid = param_grid
)

# View summary
dplyr::group_by(results, hidden, lr, activation, epochs, loss, metric) |>
  dplyr::summarise(mean = mean(value, na.rm = TRUE), .groups = "drop")
}

```

---

 ibs\_survmat

*Integrated Brier Score (IBS) from a Survival Probability Matrix*


---

## Description

Computes the Integrated Brier Score (IBS) over a set of evaluation time points, using trapezoidal integration and IPCW adjustment for right-censoring.

## Usage

```
ibs_survmat(object, sp_matrix, times)
```

**Arguments**

object	A 'Surv' object with observed time and status.
sp_matrix	A data frame or matrix of predicted survival probabilities. Each column corresponds to a time point in 'times'.
times	A numeric vector of time points. Must match the columns of 'sp_matrix'.

**Value**

A single numeric value representing the integrated Brier score.

**Examples**

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  set.seed(123)
  veteran <- survival::veteran
  idx <- sample(nrow(veteran), 0.7 * nrow(veteran))
  train <- veteran[idx, ]; test <- veteran[-idx, ]
  mod <- survdmn(survival::Surv(time, status) ~
    age + karno + celltype, data = train, epochs = 50, verbose = FALSE)
  pred <- predict(mod, newdata = test, times = c(30, 90, 180), type = "survival")
  y_test <- model.response(model.frame(mod$formula, test))
  ibs_survmat(y_test, sp_matrix = pred, times = c(30, 90, 180))
}
```

---

 plot.survdmn

*Plot survdmn Survival Curves using ggplot2*


---

**Description**

Visualizes survival curves predicted by a fitted 'survdmn' model. Curves can be grouped by a categorical variable in 'newdata' and optionally display only the group-wise means or overlay them.

**Usage**

```
## S3 method for class 'survdmn'
plot(
  x,
  newdata = NULL,
  times = 1:365,
  group_by = NULL,
  plot_mean_only = FALSE,
  add_mean = TRUE,
  alpha = 0.3,
  mean_lwd = 1.3,
  mean_lty = 1,
  ...
)
```

**Arguments**

x	A fitted 'survdnn' model object.
newdata	Optional data frame for prediction (defaults to training data).
times	A numeric vector of time points at which to compute survival probabilities.
group_by	Optional name of a column in 'newdata' used to color and group curves.
plot_mean_only	Logical; if 'TRUE', plots only the mean survival curve per group.
add_mean	Logical; if 'TRUE', adds mean curves to the individual lines.
alpha	Alpha transparency for individual curves (ignored if 'plot_mean_only = TRUE').
mean_lwd	Line width for mean survival curves.
mean_lty	Line type for mean survival curves.
...	Reserved for future use.

**Value**

A 'ggplot' object.

**Examples**

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  set.seed(42)
  veteran <- survival::veteran
  mod <- survdnn(survival::Surv(time, status) ~ age + karno + celltype, data = veteran,
                hidden = c(16, 8), epochs = 100, verbose = FALSE)
  plot(mod, group_by = "celltype", times = 1:300)
}
```

---

plot\_loss

*Plot Training Loss for a survdnn Model*

---

**Description**

Visualize the evolution of the training loss across epochs for a fitted 'survdnn' model. Helps inspect convergence, instability, or callback effects (e.g., early stopping).

**Usage**

```
plot_loss(object, smooth = FALSE, log_y = FALSE, ...)
```

**Arguments**

object	A fitted 'survdnn' model.
smooth	Logical; if 'TRUE', overlays a smoothed loess curve.
log_y	Logical; if 'TRUE', uses a log10 y-scale.
...	Reserved for future use.

**Value**

A 'ggplot' object.

---

predict.survdmn	<i>Predict from a survdmn Model</i>
-----------------	-------------------------------------

---

**Description**

Generate predictions from a fitted 'survdmn' model for new data. Supports linear predictors, survival probabilities at specified time points, or cumulative risk estimates.

**Usage**

```
## S3 method for class 'survdmn'
predict(object, newdata, times = NULL, type = c("survival", "lp", "risk"), ...)
```

**Arguments**

object	An object of class "survdmn" returned by [survdmn()].
newdata	A data frame of new observations to predict on.
times	Numeric vector of time points at which to compute survival or risk probabilities. Required if 'type = "survival"' or 'type = "risk"' for Cox/AFT models. For CoxTime, 'times = NULL' is allowed when 'type="survival"' and defaults to event times.
type	Character string specifying the type of prediction to return: <b>"lp"</b> Linear predictor. For "cox"/"cox_l2" this is a log-risk score (higher implies worse prognosis, consistent with training sign convention). For "aft", this is the predicted location parameter $\mu(x)$ on the log-time scale. For "coxtime", this is $g(t_0, x)$ evaluated at a reference time $t_0$ (the first event time). <b>"survival"</b> Predicted survival probabilities at each value of 'times'. <b>"risk"</b> Cumulative risk (1 - survival) at <b>a single</b> time point.
...	Currently ignored (for future extensions).

**Value**

A numeric vector (if 'type = "lp"' or "risk"), or a data frame (if 'type = "survival"') with one row per observation and one column per 'times'.

---

```
print.survdnn      Print a survdnn Model
```

---

**Description**

Pretty prints a fitted ‘survdnn’ model. Displays the formula, network architecture, training configuration, and final training loss.

**Usage**

```
## S3 method for class 'survdnn'
print(x, ...)
```

**Arguments**

```
x          An object of class "survdnn", returned by [survdnn()].
...        Ignored (for future compatibility).
```

**Value**

The model object, invisibly.

**Examples**

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  veteran <- survival::veteran
  mod <- survdnn(survival::Surv(time, status) ~
    age + karno + celltype, data = veteran, epochs = 20, verbose = FALSE)
  print(mod)
}
```

---

```
summarize_cv_survdnn  Summarize Cross-Validation Results from survdnn
```

---

**Description**

Computes mean, standard deviation, and confidence intervals for metrics from cross-validation.

**Usage**

```
summarize_cv_survdnn(cv_results, by_time = TRUE, conf_level = 0.95)
```

**Arguments**

`cv_results` A tibble returned by `[cv_survdnn()]`.  
`by_time` Logical. Whether to stratify results by ‘time’ (if present).  
`conf_level` Confidence level for the intervals (default: 0.95).

**Value**

A tibble summarizing mean, sd, and confidence bounds per metric (and per time if applicable).

**Examples**

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {
  veteran <- survival::veteran
  res <- cv_survdnn(
    survival::Surv(time, status) ~ age + karno + celltype,
    data = veteran,
    times = c(30, 90, 180, 270),
    metrics = c("cindex", "ibs"),
    folds = 3,
    .seed = 42,
    hidden = c(16, 8),
    epochs = 5
  )
  summarize_cv_survdnn(res)
}
```

---

summarize\_tune\_survdnn

*Summarize survdnn Tuning Results*

---

**Description**

Aggregates cross-validation results from ‘`tune_survdnn(return = "all")`’ by configuration, metric, and optionally by time point.

**Usage**

```
summarize_tune_survdnn(tuning_results, by_time = TRUE)
```

**Arguments**

`tuning_results` The full tibble returned by ‘`tune_survdnn(..., return = "all")`’.  
`by_time` Logical; whether to group and summarize separately by time points.

**Value**

A summarized tibble with mean and standard deviation of performance metrics.

---

`summary.survdsn`*Summarize a Deep Survival Neural Network Model*

---

## Description

Provides a structured summary of a fitted ‘survdsn’ model, including the network architecture, training configuration, and data characteristics. The summary is printed automatically with a styled header and sectioned output using {cli} and base formatting. The object is returned invisibly.

## Usage

```
## S3 method for class 'survdsn'  
summary(object, ...)
```

## Arguments

<code>object</code>	An object of class “survdsn” returned by the [survdsn()] function.
<code>...</code>	Currently ignored (for future compatibility).

## Value

Invisibly returns an object of class “summary.survdsn”.

## Examples

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {  
  set.seed(42)  
  sim_data <- data.frame(  
    age = rnorm(100, 60, 10),  
    sex = factor(sample(c("male", "female"), 100, TRUE)),  
    trt = factor(sample(c("A", "B"), 100, TRUE)),  
    time = rexp(100, 0.05),  
    status = rbinom(100, 1, 0.7)  
  )  
  mod <- survdsn(  
    survival::Surv(time, status) ~ age + sex + trt,  
    data = sim_data,  
    epochs = 50,  
    verbose = FALSE  
  )  
  summary(mod)  
}
```

survdnn

*Fit a Deep Neural Network for Survival Analysis***Description**

Trains a deep neural network (DNN) to model right-censored survival data using one of the predefined loss functions: Cox, AFT, or Coxtime.

**Usage**

```
survdnn(
  formula,
  data,
  hidden = c(32L, 16L),
  activation = "relu",
  lr = 1e-04,
  epochs = 300L,
  loss = c("cox", "cox_l2", "aft", "coxtime"),
  optimizer = c("adam", "adamw", "sgd", "rmsprop", "adagrad"),
  optim_args = list(),
  verbose = TRUE,
  dropout = 0.3,
  batch_norm = TRUE,
  callbacks = NULL,
  .seed = NULL,
  .device = c("auto", "cpu", "cuda"),
  .threads = NULL,
  na_action = c("omit", "fail")
)
```

**Arguments**

formula	A survival formula of the form ‘Surv(time, status) ~ predictors’.
data	A data frame containing the variables in the model.
hidden	Integer vector. Sizes of the hidden layers (default: c(32, 16)).
activation	Character string specifying the activation function to use in each layer. Supported options: "relu", "leaky_relu", "tanh", "sigmoid", "gelu", "elu", "softplus".
lr	Learning rate for the optimizer (default: '1e-4').
epochs	Number of training epochs (default: 300).
loss	Character name of the loss function to use. One of "cox", "cox_l2", "aft", or "coxtime".
optimizer	Character string specifying the optimizer to use. One of "adam", "adamw", "sgd", "rmsprop", or "adagrad". Defaults to "adam".

<code>optim_args</code>	Optional named list of additional arguments passed to the underlying torch optimizer (e.g., <code>'list(weight_decay = 1e-4, momentum = 0.9)'</code> ).
<code>verbose</code>	Logical; whether to print progress and periodic loss updates during fitting (default: <code>TRUE</code> ).
<code>dropout</code>	Numeric between 0 and 1. Dropout rate applied after each hidden layer (default = 0.3). Set to 0 to disable dropout.
<code>batch_norm</code>	Logical; whether to add batch normalization after each hidden linear layer (default = <code>TRUE</code> ).
<code>callbacks</code>	Optional list of callback functions. Each callback should have signature <code>'function(epoch, current)'</code> and return <code>TRUE</code> if training should stop, <code>FALSE</code> otherwise. Used, for example, with <code>[callback_early_stopping()]</code> .
<code>.seed</code>	Optional integer. If provided, sets both R and torch random seeds for reproducible weight initialization, shuffling, and dropout.
<code>.device</code>	Character string indicating the computation device. One of <code>"auto"</code> , <code>"cpu"</code> , or <code>"cuda"</code> . <code>"auto"</code> uses CUDA if available, otherwise falls back to CPU.
<code>.threads</code>	Optional positive integer. If provided, sets the global Torch CPU thread count via <code>'torch::torch_set_num_threads()'</code> .
<code>na_action</code>	Character. How to handle missing values in the model variables: <code>"omit"</code> drops incomplete rows (and reports how many were removed when <code>'verbose=TRUE'</code> ); <code>"fail"</code> stops with an error if any missing values are present.

## Value

An object of class `"survdnn"` containing:

**model** Trained `'nn_module'` object.

**formula** Original survival formula.

**data** Training data used for fitting.

**xnames** Predictor variable names.

**x\_center** Column means of predictors.

**x\_scale** Column standard deviations of predictors.

**loss\_history** Vector of loss values per epoch.

**final\_loss** Final training loss.

**loss** Loss function name used (`"cox"`, `"aft"`, etc.).

**activation** Activation function used.

**hidden** Hidden layer sizes.

**lr** Learning rate.

**epochs** Number of training epochs.

**optimizer** Optimizer name used.

**optim\_args** List of optimizer arguments used.

**device** Torch device used for training (`'torch_device'`).

**threads** CPU thread setting passed via `'.threads'`; `'NULL'` means Torch default/global setting.

**aft\_log\_sigma** Learned global log(sigma) for 'loss="aft"'; 'NA\_real\_' otherwise.  
**aft\_loc** AFT log-time location offset used for centering when 'loss="aft"'; 'NA\_real\_' otherwise.  
**covertime\_time\_center** Mean used to scale time for CoxTime; 'NA\_real\_' otherwise.  
**covertime\_time\_scale** SD used to scale time for CoxTime; 'NA\_real\_' otherwise.

tune\_survdmn

*Tune Hyperparameters for a survdmn Model via Cross-Validation***Description**

Performs k-fold cross-validation over a user-defined hyperparameter grid and selects the best configuration according to the specified evaluation metric.

**Usage**

```
tune_survdmn(
  formula,
  data,
  times,
  metrics = "cindex",
  param_grid,
  folds = 3,
  .seed = 42,
  .device = c("auto", "cpu", "cuda"),
  .threads = NULL,
  na_action = c("omit", "fail"),
  verbose = TRUE,
  refit = FALSE,
  return = c("all", "summary", "best_model")
)
```

**Arguments**

formula	A survival formula, e.g., 'Surv(time, status) ~ x1 + x2'.
data	A data frame.
times	A numeric vector of evaluation time points.
metrics	A character vector of evaluation metrics: "cindex", "brier", or "ibs". Only the first metric is used for model selection.
param_grid	A named list defining hyperparameter combinations to evaluate. Required names: 'hidden', 'lr', 'activation', 'epochs', 'loss'.
folds	Number of cross-validation folds (default: 3).
.seed	Optional seed for reproducibility.
.device	Character string indicating the computation device used when fitting models during cross-validation and refitting. One of "auto", "cpu", or "cuda". "auto" uses CUDA if available, otherwise falls back to CPU.

. threads	Optional positive integer. If provided, sets Torch CPU thread count for all nested model fits via <code>torch::torch_set_num_threads()</code> .
na_action	Character. How to handle missing values: <code>"omit"</code> drops incomplete rows; <code>"fail"</code> errors if any NA is present.
verbose	Logical; whether to print tuning progress and propagate verbose messages to nested cross-validation and optional refit (default: TRUE).
refit	Logical. If TRUE, refits the best model on the full dataset.
return	One of "all", "summary", or "best_model": <b>"all"</b> Returns the full cross-validation result across all combinations. <b>"summary"</b> Returns averaged results per configuration. <b>"best_model"</b> Returns the refitted model or best hyperparameters.

**Value**

A tibble or model object depending on the `'return'` value.

# Index

brier, [2](#)

callback\_early\_stopping, [3](#)

cindex\_survmat, [4](#)

cv\_survdnn, [4](#)

evaluate\_survdnn, [6](#)

gridsearch\_survdnn, [7](#)

ibs\_survmat, [8](#)

plot\_survdnn, [9](#)

plot\_loss, [10](#)

predict\_survdnn, [11](#)

print\_survdnn, [12](#)

summarize\_cv\_survdnn, [12](#)

summarize\_tune\_survdnn, [13](#)

summary\_survdnn, [14](#)

survdnn, [15](#)

tune\_survdnn, [17](#)