Package 'daltoolboxdp'

May 22, 2026

Title Deep Python Extensions for 'daltoolbox'

Version 1.3.747

Description Extends 'daltoolbox' with Python-backed components for deep learning, scikit-learn classification, and time-series forecasting through 'reticulate'. The package provides objects that follow the 'daltoolbox' architecture while delegating model creation, fitting, encoding, and prediction to Python libraries such as 'torch' and 'scikit-learn'. In the package name, 'dp' stands for 'Deep Python'. The overall workflow is inspired by the Experiment Lines approach described in Ogasawara et al. (2009) <doi:10.1007/978-3-642-02279-1_20>.

License MIT + file LICENSE

URL https://cefet-rj-dal.github.io/daltoolboxdp/, https://github.com/cefet-rj-dal/daltoolboxdp

BugReports https://github.com/cefet-rj-dal/daltoolboxdp/issues

Encoding UTF-8

RoxygenNote 8.0.0

Depends R (>= 4.1.0)

Imports tspredit, daltoolbox, reticulate

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/reticulate list(packages = list(list(package = ``scipy''), list(package = ``torch''), list(package = ``pandas''), list(package = ``numpy''), list(package = ``matplotlib''), list(package = ``scikit-learn'')))

NeedsCompilation no

Author Eduardo Ogasawara [aut, ths, cre] (ORCID: <https://orcid.org/0000-0002-0466-0626>),
Diego Salles [aut],
Erich Carvalho [aut],
Janio Lima [aut],
Joao Kongevold [aut],

Lucas Tavares [aut],
Eduardo Bezerra [ctb],
CEFET/RJ [cph]

Maintainer Eduardo Ogasawara <eogasawara@ieee.org>

Repository CRAN

Date/Publication 2026-05-22 12:40:09 UTC

Contents

`autoenc_adv_e` *Adversarial Autoencoder - Encode*

Description

Creates an adversarial autoencoder (AAE) with configurable encoder, decoder and discriminator topologies through a Python/PyTorch backend.

Usage

```
autoenc_adv_e(
  input_size,
  encoding_size,
  encoder_hidden_sizes = c(60L, 60L),
  decoder_hidden_sizes = c(60L, 60L),
  discriminator_hidden_sizes = c(60L, 60L),
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  dropout = 0.4,
  latent_prior_scale = 5,
  lr_encoder = NULL,
  lr_decoder = NULL,
  lr_generator = NULL,
  lr_discriminator = NULL,
  batch_size = 350,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

`input_size`	Integer. Number of input features per observation.
`encoding_size`	Integer. Size of the latent (bottleneck) representation.
`encoder_hidden_sizes`	Integer vector used by the encoder network. Default is `c(60L, 60L)`.
`decoder_hidden_sizes`	Integer vector used by the decoder network. Default is `c(60L, 60L)`.
`discriminator_hidden_sizes`	Integer vector used by the discriminator network. Default is `c(60L, 60L)`.
`activation`	Character. Hidden activation function. One of `"relu"`, `"leaky_relu"`, `"elu"`, `"gelu"`, or `"tanh"`.
`dropout`	Numeric. Dropout rate applied to adversarial hidden layers.
`latent_prior_scale`	Numeric. Standard deviation scale used to sample the latent prior.
`lr_encoder`	Optional numeric. Learning rate of the encoder reconstruction optimizer.
`lr_decoder`	Optional numeric. Learning rate of the decoder reconstruction optimizer.

lr_generator	Optional numeric. Learning rate of the encoder adversarial optimizer.
lr_discriminator	Optional numeric. Learning rate of the discriminator optimizer.
batch_size	Integer. Mini-batch size used during training. Default is 350.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Base optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

The adversarial autoencoder exposes the latent prior scale, dropout, activation family, and optimizer learning rates for each adversarial component. If the component-specific learning rates are left as NULL, the wrapper derives them from learning_rate using the training defaults of the Python implementation.

Value

A autoenc_adv_e object.

References

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2016). Adversarial Autoencoders.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_adv_e(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L),
  discriminator_hidden_sizes = c(64L, 32L),
  latent_prior_scale = 2
)
ae <- daltoolbox::fit(ae, X)
Z <- daltoolbox::transform(ae, X)
```

```
## End(Not run)
```

`autoenc_adv_ed` *Adversarial Autoencoder - Encode-Decode*

Description

Creates an adversarial autoencoder (AAE) that reconstructs observations while regularizing the latent space through a discriminator, using a Python/PyTorch backend.

Usage

```
autoenc_adv_ed(
  input_size,
  encoding_size,
  encoder_hidden_sizes = c(60L, 60L),
  decoder_hidden_sizes = c(60L, 60L),
  discriminator_hidden_sizes = c(60L, 60L),
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  dropout = 0.4,
  latent_prior_scale = 5,
  lr_encoder = NULL,
  lr_decoder = NULL,
  lr_generator = NULL,
  lr_discriminator = NULL,
  batch_size = 350,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

`input_size`	Integer. Number of input features per observation.
`encoding_size`	Integer. Size of the latent (bottleneck) representation.

`encoder_hidden_sizes`
: Integer vector used by the encoder network. Default is `c(60L, 60L)`.

`decoder_hidden_sizes`
: Integer vector used by the decoder network. Default is `c(60L, 60L)`.

`discriminator_hidden_sizes`
: Integer vector used by the discriminator network. Default is `c(60L, 60L)`.

`activation`
: Character. Hidden activation function. One of `"relu"`, `"leaky_relu"`, `"elu"`, `"gelu"`, or `"tanh"`.

`dropout`
: Numeric. Dropout rate applied to adversarial hidden layers.

`latent_prior_scale`
: Numeric. Standard deviation scale used to sample the latent prior.

`lr_encoder`
: Optional numeric. Learning rate of the encoder reconstruction optimizer.

`lr_decoder`
: Optional numeric. Learning rate of the decoder reconstruction optimizer.

`lr_generator`
: Optional numeric. Learning rate of the encoder adversarial optimizer.

`lr_discriminator`
: Optional numeric. Learning rate of the discriminator optimizer.

`batch_size`
: Integer. Mini-batch size used during training. Default is 350.

`epochs`
: Integer. Maximum number of training epochs. Default is 100.

`num_epochs`
: Deprecated compatibility alias for `epochs`. If informed, it overrides `epochs`.

`learning_rate`
: Numeric. Base optimizer learning rate. Default is 0.001.

`validation_strategy`
: Character. One of `static` or `dynamic`.

`stopping_rule`
: Character. One of `none`, `patience`, `sma`, `ema`, or `h`.

`val_ratio`
: Numeric. Validation fraction used when validation is enabled. Default is 0.3.

`patience`
: Integer. Early stopping patience. Default is 100.

`min_delta`
: Numeric. Minimum improvement to reset early stopping. Default is 1e-4.

`sma_window`
: Integer. Window size used by `sma`. Default is 5.

`ema_alpha`
: Numeric. Smoothing factor used by `ema`. Default is 0.2.

`test_window`
: Integer. Window size used by `h`. Default is 30.

`p_value`
: Numeric. Significance threshold used by `h`. Default is 0.05.

Value

A `autoenc_adv_ed` object.

References

Makhzani, A. et al. (2016). Adversarial Autoencoders.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_adv_ed(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L),
  discriminator_hidden_sizes = c(64L, 32L),
  latent_prior_scale = 2
)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)

## End(Not run)
```

---

autoenc_conv_e *Convolutional Autoencoder - Encode*

Description

Creates a deep learning convolutional autoencoder (ConvAE) to encode sequences of observations. Wraps a PyTorch implementation.

Usage

```
autoenc_conv_e(
  input_size,
  encoding_size,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.

batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Value

A autoenc_conv_e object.

References

Masci, J., Meier, U., Ciresan, D., & Schmidhuber, J. (2011). Stacked Convolutional Auto-Encoders.

Examples

```
## Not run:
# Conv1D-based encoder expects data reshaped internally to (n, input_size, 1)
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_conv_e(input_size = 20, encoding_size = 5, epochs = 100)
ae <- daltoolbox::fit(ae, X)
Z  <- daltoolbox::transform(ae, X)   # 50 x 5 encodings

## End(Not run)

# See:
# https://github.com/cefet-rj-dal/daltoolbox/blob/main/autoencoder/autoenc_conv_e.md
```

`autoenc_conv_ed` *Convolutional Autoencoder - Encode-Decode*

Description

Creates a deep learning convolutional autoencoder (ConvAE) to encode and decode sequences of observations. Wraps a PyTorch implementation.

Usage

```
autoenc_conv_ed(
  input_size,
  encoding_size,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

`input_size`	Integer. Number of input features per observation.
`encoding_size`	Integer. Size of the latent (bottleneck) representation.
`batch_size`	Integer. Mini-batch size used during training. Default is 32.
`epochs`	Integer. Maximum number of training epochs. Default is 100.
`num_epochs`	Deprecated compatibility alias for `epochs`. If informed, it overrides `epochs`.
`learning_rate`	Numeric. Optimizer learning rate. Default is 0.001.
`validation_strategy`	Character. One of `static` or `dynamic`.
`stopping_rule`	Character. One of `none`, `patience`, `sma`, `ema`, or `h`.
`val_ratio`	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
`patience`	Integer. Early stopping patience. Default is 100.
`min_delta`	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
`sma_window`	Integer. Window size used by `sma`. Default is 5.

ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Value

A autoenc_conv_ed object.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_conv_ed(input_size = 20, encoding_size = 5, epochs = 100)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)  # same dims as X
mean((X - X_hat)^2)

## End(Not run)

# See:
# https://github.com/cefet-rj-dal/daltoolbox/blob/main/autoencoder/autoenc_conv_ed.md
```

---

autoenc_denoise_e *Denoising Autoencoder - Encode*

Description

Creates a denoising autoencoder that learns robust latent representations from corrupted inputs through a Python/PyTorch backend.

Usage

```
autoenc_denoise_e(
  input_size,
  encoding_size,
  encoder_hidden_sizes = 64L,
  decoder_hidden_sizes = NULL,
  activation = c("relu", "leaky_relu", "elu", "gelu", "selu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  negative_slope = 0.2,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  noise_factor = 0.3,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
```

```
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
encoder_hidden_sizes	Integer vector. Hidden sizes used by the encoder.
decoder_hidden_sizes	Optional integer vector. Hidden sizes used by the decoder. If NULL, the decoder mirrors encoder_hidden_sizes in reverse order.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", "selu", or "tanh".
output_activation	Character. Output activation of the decoder. One of "none", "relu", "sigmoid", "tanh", or "softplus".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
noise_factor	Numeric. Standard deviation (scale) of the noise added during training.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

Besides the denoising factor, this constructor exposes the same encoder/decoder customization available in autoenc_e(). This allows the user to combine shallow or deep dense architectures with stochastic input corruption.

Value

A autoenc_denoise_e object.

References

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_denoise_e(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L),
  noise_factor = 0.2
)
ae <- daltoolbox::fit(ae, X)
Z <- daltoolbox::transform(ae, X)
dim(Z)

## End(Not run)
```

---

autoenc_denoise_ed *Denoising Autoencoder - Encode-Decode*

Description

Creates a denoising autoencoder that reconstructs observations after learning from corrupted inputs through a Python/PyTorch backend.

Usage

```
autoenc_denoise_ed(
  input_size,
  encoding_size,
  encoder_hidden_sizes = 64L,
  decoder_hidden_sizes = NULL,
  activation = c("relu", "leaky_relu", "elu", "gelu", "selu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  negative_slope = 0.2,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  noise_factor = 0.3,
```

```
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
encoder_hidden_sizes	Integer vector. Hidden sizes used by the encoder.
decoder_hidden_sizes	Optional integer vector. Hidden sizes used by the decoder. If NULL, the decoder mirrors encoder_hidden_sizes in reverse order.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", "selu", or "tanh".
output_activation	Character. Output activation of the decoder. One of "none", "relu", "sigmoid", "tanh", or "softplus".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
noise_factor	Numeric. Standard deviation (scale) of the noise added during training.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Value

A autoenc_denoise_ed object.

References

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_denoise_ed(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L),
  noise_factor = 0.2
)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)
mean((X - X_hat)^2)

## End(Not run)
```

---

autoenc_e *Autoencoder - Encode*

Description

Creates a dense autoencoder that learns a latent representation for a sequence of observations through a Python/PyTorch backend.

Usage

```
autoenc_e(
  input_size,
  encoding_size,
  encoder_hidden_sizes = 64L,
  decoder_hidden_sizes = NULL,
  activation = c("relu", "leaky_relu", "elu", "gelu", "selu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  negative_slope = 0.2,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
```

```
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
encoder_hidden_sizes	Integer vector. Hidden sizes used by the encoder. Default is 64L, matching the previous implementation.
decoder_hidden_sizes	Optional integer vector. Hidden sizes used by the decoder. If NULL, the decoder mirrors encoder_hidden_sizes in reverse order.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", "selu", or "tanh".
output_activation	Character. Output activation of the decoder. One of "none", "relu", "sigmoid", "tanh", or "softplus".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

The dense autoencoder is now architecture-configurable. You can keep the original single hidden layer with encoder_hidden_sizes = 64, or define deeper asymmetric encoder/decoder stacks such as encoder_hidden_sizes = c(128L, 64L, 32L) and decoder_hidden_sizes = c(32L, 64L, 128L).

Value

A autoenc_e object.

References

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library.

Examples

```
## Not run:
X <- matrix(rnorm(2000), nrow = 100, ncol = 20)

ae <- autoenc_e(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L),
  activation = "relu"
)
ae <- daltoolbox::fit(ae, X)
Z <- daltoolbox::transform(ae, X)
dim(Z)

## End(Not run)
```

---

autoenc_ed *Autoencoder - Encode-Decode*

Description

Creates a dense autoencoder that compresses and reconstructs observations through a Python/PyTorch backend.

Usage

```
autoenc_ed(
  input_size,
  encoding_size,
  encoder_hidden_sizes = 64L,
  decoder_hidden_sizes = NULL,
  activation = c("relu", "leaky_relu", "elu", "gelu", "selu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  negative_slope = 0.2,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
```

```
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
encoder_hidden_sizes	Integer vector. Hidden sizes used by the encoder. Default is 64L, matching the previous implementation.
decoder_hidden_sizes	Optional integer vector. Hidden sizes used by the decoder. If NULL, the decoder mirrors encoder_hidden_sizes in reverse order.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", "selu", or "tanh".
output_activation	Character. Output activation of the decoder. One of "none", "relu", "sigmoid", "tanh", or "softplus".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

This variant exposes the same architecture controls as `autoenc_e()`, but the transformation returns reconstructions in the original input space. Use it when reconstruction quality is part of the analysis or when the autoencoder is used for anomaly detection.

Value

A `autoenc_ed` object.

References

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_ed(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L),
  decoder_hidden_sizes = c(64L, 128L)
)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)
mean((X - X_hat)^2)

## End(Not run)
```

`autoenc_lstm_e` *LSTM Autoencoder - Encode*

Description

Creates an LSTM-based autoencoder with configurable recurrent depth and latent projection through a Python/PyTorch backend.

Usage

```
autoenc_lstm_e(
  input_size,
  encoding_size,
  lstm_hidden_size = NULL,
  sequence_length = 1L,
  num_layers = 1L,
```

```
  dropout = 0,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
lstm_hidden_size	Optional integer. Hidden size used inside the encoder/decoder LSTMs. If NULL, it defaults to encoding_size.
sequence_length	Integer. Number of time steps represented by each row. input_size must be divisible by sequence_length. Default is 1L, which preserves the previous behavior.
num_layers	Integer. Number of recurrent LSTM layers.
dropout	Numeric. Recurrent dropout applied between LSTM layers when num_layers > 1.
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

encoding_size remains the latent bottleneck exposed to the user. The recurrent body can now use a different lstm_hidden_size, multiple layers, dropout between recurrent layers, and a configurable sequence_length to reshape each row into a sequence before encoding.

Value

A autoenc_lstm_e object.

References

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_lstm_e(
  input_size = 20,
  encoding_size = 5,
  lstm_hidden_size = 16,
  sequence_length = 4,
  num_layers = 2,
  dropout = 0.1
)
ae <- daltoolbox::fit(ae, X)
Z <- daltoolbox::transform(ae, X)
dim(Z)

## End(Not run)
```

---

autoenc_lstm_ed *LSTM Autoencoder - Encode-Decode*

Description

Creates an LSTM-based autoencoder that reconstructs observations after sequence-aware compression through a Python/PyTorch backend.

Usage

```
autoenc_lstm_ed(
  input_size,
  encoding_size,
  lstm_hidden_size = NULL,
  sequence_length = 1L,
  num_layers = 1L,
```

```
  dropout = 0,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
lstm_hidden_size	Optional integer. Hidden size used inside the encoder/decoder LSTMs. If NULL, it defaults to encoding_size.
sequence_length	Integer. Number of time steps represented by each row. input_size must be divisible by sequence_length. Default is 1L, which preserves the previous behavior.
num_layers	Integer. Number of recurrent LSTM layers.
dropout	Numeric. Recurrent dropout applied between LSTM layers when num_layers > 1.
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Value

A autoenc_lstm_ed object.

References

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_lstm_ed(
  input_size = 20,
  encoding_size = 5,
  lstm_hidden_size = 16,
  sequence_length = 4,
  num_layers = 2,
  dropout = 0.1
)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)

## End(Not run)
```

---

autoenc_stacked_e *Stacked Autoencoder - Encode*

Description

Creates a stacked autoencoder with stage-wise configurable latent sizes and dense sub-architectures through a Python/PyTorch backend.

Usage

```
autoenc_stacked_e(
  input_size,
  encoding_size,
  encoding_sizes = NULL,
  encoder_hidden_sizes = 64L,
  decoder_hidden_sizes = NULL,
  activation = c("relu", "leaky_relu", "elu", "gelu", "selu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  negative_slope = 0.2,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
```

```
  k = 3,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Default latent size used when encoding_sizes = NULL.
encoding_sizes	Optional integer vector. Stage-specific latent sizes. If supplied, it defines the number of stages and overrides k.
encoder_hidden_sizes	Integer vector shared by all stages, or a list of integer vectors with one encoder architecture per stage.
decoder_hidden_sizes	Optional integer vector or list mirroring encoder_hidden_sizes. If NULL, each stage mirrors its encoder in reverse order.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", "selu", or "tanh".
output_activation	Character. Output activation used by stage decoders. One of "none", "relu", "sigmoid", "tanh", or "softplus".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
k	Integer. Number of stacked stages when encoding_sizes = NULL.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

The stacked autoencoder now supports progressively different latent widths across stages. Keep `k = 3` and `encoding_sizes = NULL` to repeat the original bottleneck, or use `encoding_sizes = c(16L, 8L, 4L)` to progressively compress the representation. `encoder_hidden_sizes` and `decoder_hidden_sizes` may be either a single integer vector shared by all stages or an R list with one integer vector per stage.

Value

A `autoenc_stacked_e` object.

References

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_stacked_e(
  input_size = 20,
  encoding_size = 5,
  encoding_sizes = c(12L, 8L, 5L),
  encoder_hidden_sizes = list(c(64L), c(32L), c(16L))
)
ae <- daltoolbox::fit(ae, X)
Z <- daltoolbox::transform(ae, X)

## End(Not run)
```

---

autoenc_stacked_ed *Stacked Autoencoder - Encode-Decode*

Description

Creates a stacked autoencoder that compresses through multiple stages and reconstructs back to the original input space through a Python/PyTorch backend.

Usage

```
autoenc_stacked_ed(
  input_size,
  encoding_size,
  encoding_sizes = NULL,
  encoder_hidden_sizes = 64L,
```

```
  decoder_hidden_sizes = NULL,
  activation = c("relu", "leaky_relu", "elu", "gelu", "selu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  negative_slope = 0.2,
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  k = 3,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Default latent size used when encoding_sizes = NULL.
encoding_sizes	Optional integer vector. Stage-specific latent sizes. If supplied, it defines the number of stages and overrides k.
encoder_hidden_sizes	Integer vector shared by all stages, or a list of integer vectors with one encoder architecture per stage.
decoder_hidden_sizes	Optional integer vector or list mirroring encoder_hidden_sizes. If NULL, each stage mirrors its encoder in reverse order.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", "selu", or "tanh".
output_activation	Character. Output activation used by stage decoders. One of "none", "relu", "sigmoid", "tanh", or "softplus".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
k	Integer. Number of stacked stages when encoding_sizes = NULL.
validation_strategy	Character. One of static or dynamic.

stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Value

A autoenc_stacked_ed object.

References

Vincent, P. et al. (2010). Stacked Denoising Autoencoders.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_stacked_ed(
  input_size = 20,
  encoding_size = 5,
  encoding_sizes = c(12L, 8L, 5L),
  encoder_hidden_sizes = list(c(64L), c(32L), c(16L))
)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)

## End(Not run)
```

---

autoenc_variational_e *Variational Autoencoder - Encode*

Description

Creates a variational autoencoder (VAE) with configurable dense encoder/decoder blocks through a Python/PyTorch backend.

Usage

```
autoenc_variational_e(
  input_size,
  encoding_size,
  encoder_hidden_sizes = c(64L, 32L),
  decoder_hidden_sizes = NULL,
  activation = c("leaky_relu", "relu", "elu", "gelu", "tanh"),
  negative_slope = 0.2,
  output_activation = c("sigmoid", "none", "relu", "tanh", "softplus"),
  reconstruction_loss = c("bce", "mse"),
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

input_size	Integer. Number of input features per observation.
encoding_size	Integer. Size of the latent (bottleneck) representation.
encoder_hidden_sizes	Integer vector used by the encoder backbone. Default is c(64L, 32L), matching the previous implementation.
decoder_hidden_sizes	Optional integer vector used by the decoder backbone. If NULL, the decoder mirrors encoder_hidden_sizes in reverse order.
activation	Character. Hidden activation function. One of "leaky_relu", "relu", "elu", "gelu", or "tanh".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
output_activation	Character. Output activation of the decoder. One of "sigmoid", "none", "relu", "tanh", or "softplus".
reconstruction_loss	Character. Reconstruction term used in the ELBO. One of "bce" or "mse".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.

learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Details

The VAE now exposes the hidden layout of both encoder and decoder, the activation family, the latent reconstruction head, and the reconstruction loss. This makes it possible to move from the original 64 -> 32 -> latent structure to deeper or shallower alternatives.

Value

A autoenc_variational_e object.

References

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_variational_e(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L, 32L),
  reconstruction_loss = "mse"
)
ae <- daltoolbox::fit(ae, X)
Z <- daltoolbox::transform(ae, X)
dim(Z)

## End(Not run)
```

`autoenc_variational_ed` *Variational Autoencoder - Encode-Decode*

Description

Creates a variational autoencoder (VAE) that reconstructs observations from a probabilistic latent space through a Python/PyTorch backend.

Usage

```
autoenc_variational_ed(
  input_size,
  encoding_size,
  encoder_hidden_sizes = c(64L, 32L),
  decoder_hidden_sizes = NULL,
  activation = c("leaky_relu", "relu", "elu", "gelu", "tanh"),
  negative_slope = 0.2,
  output_activation = c("sigmoid", "none", "relu", "tanh", "softplus"),
  reconstruction_loss = c("bce", "mse"),
  batch_size = 32,
  epochs = 100L,
  num_epochs = NULL,
  learning_rate = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.3,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

`input_size`	Integer. Number of input features per observation.
`encoding_size`	Integer. Size of the latent (bottleneck) representation.
`encoder_hidden_sizes`	Integer vector used by the encoder backbone. Default is `c(64L, 32L)`, matching the previous implementation.
`decoder_hidden_sizes`	Optional integer vector used by the decoder backbone. If `NULL`, the decoder mirrors `encoder_hidden_sizes` in reverse order.

activation	Character. Hidden activation function. One of "leaky_relu", "relu", "elu", "gelu", or "tanh".
negative_slope	Numeric. Negative slope used when activation = "leaky_relu".
output_activation	Character. Output activation of the decoder. One of "sigmoid", "none", "relu", "tanh", or "softplus".
reconstruction_loss	Character. Reconstruction term used in the ELBO. One of "bce" or "mse".
batch_size	Integer. Mini-batch size used during training. Default is 32.
epochs	Integer. Maximum number of training epochs. Default is 100.
num_epochs	Deprecated compatibility alias for epochs. If informed, it overrides epochs.
learning_rate	Numeric. Optimizer learning rate. Default is 0.001.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled. Default is 0.3.
patience	Integer. Early stopping patience. Default is 100.
min_delta	Numeric. Minimum improvement to reset early stopping. Default is 1e-4.
sma_window	Integer. Window size used by sma. Default is 5.
ema_alpha	Numeric. Smoothing factor used by ema. Default is 0.2.
test_window	Integer. Window size used by h. Default is 30.
p_value	Numeric. Significance threshold used by h. Default is 0.05.

Value

A autoenc_variational_ed object.

References

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes.

Examples

```
## Not run:
X <- matrix(rnorm(1000), nrow = 50, ncol = 20)
ae <- autoenc_variational_ed(
  input_size = 20,
  encoding_size = 5,
  encoder_hidden_sizes = c(128L, 64L, 32L),
  reconstruction_loss = "mse"
)
ae <- daltoolbox::fit(ae, X)
X_hat <- daltoolbox::transform(ae, X)

## End(Not run)
```

skcla_gb *Gradient Boosting Classifier*

Description

Implements a classifier using the Gradient Boosting algorithm. Wraps scikit-learn's `GradientBoostingClassifier` through `reticulate`.

Usage

```
skcla_gb(
  attribute,
  slevels,
  n_estimators = 100,
  learning_rate = 0.1,
  max_depth = 3,
  subsample = 1,
  min_samples_split = 2,
  min_samples_leaf = 1,
  loss = c("log_loss", "exponential")
)
```

Arguments

`attribute`	Target attribute name for model building.
`slevels`	Possible values for the target classification.
`n_estimators`	Number of boosting stages to perform.
`learning_rate`	Learning rate that shrinks the contribution of each tree.
`max_depth`	Maximum depth of the individual regression estimators.
`subsample`	Fraction of samples used to fit each stage.
`min_samples_split`	Minimum number of samples required to split an internal node.
`min_samples_leaf`	Minimum number of samples required to be at a leaf node.
`loss`	Loss function to be optimized. One of `"log_loss"` or `"exponential"`.

Details

Tree Boosting

Value

A `skcla_gb` classifier object.

References

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine.

Examples

```
## Not run:
data(iris)
clf <- skcla_gb(
  attribute = "Species",
  slevels = levels(iris$Species),
  n_estimators = 150,
  learning_rate = 0.05
)
clf <- daltoolbox::fit(clf, iris)
pred <- predict(clf, iris)
table(pred, iris$Species)

## End(Not run)
```

---

skcla_knn *K-Nearest Neighbors Classifier*

Description

Implements classification using the k-Nearest Neighbors algorithm. Wraps scikit-learn's `KNeighborsClassifier` through `reticulate`.

Usage

```
skcla_knn(
  attribute,
  slevels,
  n_neighbors = 5,
  weights = c("uniform", "distance"),
  metric = c("euclidean", "manhattan", "chebyshev", "minkowski")
)
```

Arguments

`attribute`	Target attribute name for model building.
`slevels`	List of possible values for classification target.
`n_neighbors`	Number of neighbors to use for queries.
`weights`	Weight function used in prediction. One of `"uniform"` or `"distance"`.
`metric`	Distance metric used by the neighbor search. One of `"euclidean"`, `"manhattan"`, `"chebyshev"`, or `"minkowski"`.

Details

K-Nearest Neighbors Classifier

Value

A skcla_knn classifier object.

References

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification.

Examples

```
## Not run:
data(iris)
clf <- skcla_knn(
  attribute = "Species",
  slevels = levels(iris$Species),
  n_neighbors = 7,
  weights = "distance"
)
clf <- daltoolbox::fit(clf, iris)
pred <- predict(clf, iris)
table(pred, iris$Species)

## End(Not run)
```

---

skcla_mlp *Multi-layer Perceptron Classifier*

Description

Implements classification using a multi-layer perceptron (MLP). Wraps scikit-learn's MLPClassifier through reticulate.

Usage

```
skcla_mlp(
  attribute,
  slevels,
  hidden_layer_sizes = c(100),
  activation = c("relu", "identity", "logistic", "tanh"),
  solver = c("adam", "lbfgs", "sgd"),
  alpha = 1e-04,
  batch_size = "auto",
  learning_rate_init = 0.001,
  max_iter = 200,
  early_stopping = FALSE
)
```

Arguments

attribute	Target attribute name for model building.
slevels	List of possible values for classification target.
hidden_layer_sizes	Number of neurons in each hidden layer.
activation	Activation function for hidden layers. One of "relu", "identity", "logistic", or "tanh".
solver	Optimizer used for training. One of "adam", "lbfgs", or "sgd".
alpha	L2 penalty (regularization term).
batch_size	Size of minibatches for stochastic optimizers. Use "auto" or an integer.
learning_rate_init	Initial learning rate used by stochastic solvers.
max_iter	Maximum number of iterations.
early_stopping	Whether to use early stopping.

Details

Neural Network Classifier

Value

A skcla_mlp classifier object.

References

Bishop, C. M. (1995). Neural Networks for Pattern Recognition.

Examples

```
## Not run:
data(iris)
clf <- skcla_mlp(
  attribute = "Species",
  slevels = levels(iris$Species),
  hidden_layer_sizes = c(32, 16),
  activation = "relu"
)
clf <- daltoolbox::fit(clf, iris)
pred <- predict(clf, iris)
table(pred, iris$Species)

## End(Not run)
```

skcla_nb *Gaussian Naive Bayes Classifier*

Description

Implements classification using Gaussian Naive Bayes. Wraps scikit-learn's GaussianNB through reticulate.

Usage

```
skcla_nb(attribute, slevels, var_smoothing = 1e-09)
```

Arguments

attribute	Target attribute name for model building
slevels	List of possible values for classification target
var_smoothing	Portion of the largest variance of all features that is added to variances

Details

Naive Bayes Classifier

Value

A skcla_nb classifier object.

References

Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. (Gaussian Naive Bayes)

Examples

```
## Not run:
data(iris)

# Gaussian Naive Bayes for multi-class iris
clf <- skcla_nb(attribute = 'Species', slevels = levels(iris$Species))
clf <- daltoolbox::fit(clf, iris)
pred <- predict(clf, iris)
table(pred, iris$Species)

## End(Not run)

# More examples:
# https://github.com/cefet-rj-dal/daltoolboxdp/blob/main/examples/skcla_nb.md
```

skcla_rf *Random Forest Classifier*

Description

Implements classification using the Random Forest algorithm. Wraps scikit-learn's `RandomForestClassifier` through `reticulate`.

Usage

```
skcla_rf(
  attribute,
  slevels,
  n_estimators = 100,
  max_depth = NULL,
  min_samples_split = 2,
  min_samples_leaf = 1,
  max_features = "sqrt",
  class_weight = NULL
)
```

Arguments

`attribute`	Target attribute name for model building.
`slevels`	List of possible values for classification target.
`n_estimators`	Number of trees in the forest.
`max_depth`	Maximum tree depth value.
`min_samples_split`	Minimum samples needed for an internal node split.
`min_samples_leaf`	Minimum samples needed at a leaf node.
`max_features`	Number of features to consider at each split. Use `"sqrt"`, `"log2"`, `NULL`, or a numeric value.
`class_weight`	Optional weights associated with classes.

Details

Tree Ensemble

Value

A `skcla_rf` classifier object.

References

Breiman, L. (2001). Random Forests. Machine Learning.

Examples

```
## Not run:
data(iris)
clf <- skcla_rf(
  attribute = "Species",
  slevels = levels(iris$Species),
  n_estimators = 200,
  max_features = "sqrt"
)
clf <- daltoolbox::fit(clf, iris)
pred <- predict(clf, iris)
table(pred, iris$Species)

## End(Not run)
```

---

skcla_svc *Support Vector Machine Classification*

Description

Implements classification using support vector machines. Wraps scikit-learn's SVC through `reticulate`.

Usage

```
skcla_svc(
  attribute,
  slevels,
  C = 1,
  kernel = c("rbf", "linear", "poly", "sigmoid"),
  gamma = "scale",
  degree = 3,
  coef0 = 0,
  probability = FALSE,
  class_weight = NULL
)
```

Arguments

`attribute`	Target attribute name for model building.
`slevels`	List of possible values for classification target.
`C`	Regularization strength parameter.
`kernel`	Kernel function type. One of `"rbf"`, `"linear"`, `"poly"`, or `"sigmoid"`.
`gamma`	Kernel coefficient value. Use `"scale"`, `"auto"`, or a numeric value.
`degree`	Polynomial degree when using `kernel = "poly"`.
`coef0`	Independent term value in polynomial and sigmoid kernels.
`probability`	Whether to enable probability estimates.
`class_weight`	Optional weights associated with classes.

Details

SVM Classifier

Value

A skcla_svc classifier object.

References

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks.

Examples

```
## Not run:
data(iris)
clf <- skcla_svc(
  attribute = "Species",
  slevels = levels(iris$Species),
  kernel = "rbf",
  C = 1
)
clf <- daltoolbox::fit(clf, iris)
pred <- predict(clf, iris)
table(pred, iris$Species)

## End(Not run)
```

---

torch_cla_mlp *PyTorch MLP Classifier*

Description

Classification model backed by a configurable PyTorch MLP with unified training strategies.

Usage

```
torch_cla_mlp(
  attribute,
  slevels,
  preprocess = NA,
  input_size,
  hidden_sizes,
  num_classes = length(slevels),
  dropout = 0,
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  normalization = c("none", "batch", "layer"),
 init_method = c("default", "xavier_uniform", "xavier_normal", "kaiming_uniform",
    "kaiming_normal"),
```

```
  epochs = 100L,
  lr = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.2,
  batch_size = 64L,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05,
  weight_decay = 0
)
```

Arguments

attribute	Target attribute name.
slevels	Vector with valid class labels.
preprocess	Optional preprocessing object.
input_size	Integer. Number of input attributes.
hidden_sizes	Integer vector with hidden layer sizes.
num_classes	Integer. Number of classes. Defaults to length(slevels).
dropout	Numeric. Dropout rate.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", or "tanh".
normalization	Character. Optional normalization after each hidden linear layer. One of "none", "batch", or "layer".
init_method	Character. Weight initialization strategy. One of "default", "xavier_uniform", "xavier_normal", "kaiming_uniform", or "kaiming_normal".
epochs	Integer. Maximum number of epochs. Default is 100L.
lr	Numeric. Learning rate.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled.
batch_size	Integer. Mini-batch size.
patience	Integer. Early stopping patience.
min_delta	Numeric. Minimum improvement to reset early stopping.
sma_window	Integer. Window size used by sma.
ema_alpha	Numeric. Smoothing factor used by ema.
test_window	Integer. Window size used by h.
p_value	Numeric. Significance threshold used by h.
weight_decay	Numeric. L2 regularization.

Examples

```
## Not run:
library(daltoolboxdp)
model <- torch_cla_mlp(
  attribute = "class",
  slevels = c("A", "B"),
  input_size = 10,
  hidden_sizes = c(64L, 32L),
  normalization = "batch",
  init_method = "kaiming_uniform",
  epochs = 1000L
)

## End(Not run)
```

---

torch_reg_mlp *PyTorch MLP Regressor*

Description

Regression model backed by a configurable PyTorch MLP with unified training strategies.

Usage

```
torch_reg_mlp(
  attribute,
  preprocess = NA,
  input_size = NA,
  hidden_sizes,
  dropout = 0,
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  normalization = c("none", "batch", "layer"),
 init_method = c("default", "xavier_uniform", "xavier_normal", "kaiming_uniform",
    "kaiming_normal"),
  epochs = 100L,
  lr = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.2,
  batch_size = 64L,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

attribute	Target attribute name.
preprocess	Optional preprocessing object.
input_size	Optional integer. Number of input attributes. When omitted, it is inferred from the training data and validated against the learned predictor set.
hidden_sizes	Integer vector with hidden layer sizes.
dropout	Numeric. Dropout rate.
activation	Character. Hidden activation function. One of "relu", "leaky_relu", "elu", "gelu", or "tanh".
output_activation	Character. Output activation of the regressor head. One of "none", "relu", "sigmoid", "tanh", or "softplus".
normalization	Character. Optional normalization after each hidden linear layer. One of "none", "batch", or "layer".
init_method	Character. Weight initialization strategy. One of "default", "xavier_uniform", "xavier_normal", "kaiming_uniform", or "kaiming_normal".
epochs	Integer. Maximum number of epochs. Default is 100L.
lr	Numeric. Learning rate.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled.
batch_size	Integer. Mini-batch size.
patience	Integer. Early stopping patience.
min_delta	Numeric. Minimum improvement to reset early stopping.
sma_window	Integer. Window size used by sma.
ema_alpha	Numeric. Smoothing factor used by ema.
test_window	Integer. Window size used by h.
p_value	Numeric. Significance threshold used by h.

Examples

```
## Not run:
library(daltoolboxdp)
model <- torch_reg_mlp(
  attribute = "target",
  hidden_sizes = c(64L, 32L),
  normalization = "layer",
  output_activation = "none",
  epochs = 1000L
)

## End(Not run)
```

`torch_ts_mlp` *PyTorch Time-Series MLP*

Description

Time-series forecaster using a configurable feedforward PyTorch MLP with unified training strategies and a Python backend.

Usage

```
torch_ts_mlp(
  preprocess = NA,
  input_size = NA,
  input_map = tspredit::ts_lagmap(),
  hidden_sizes = c(16L, 8L),
  dropout = 0,
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  output_activation = c("none", "relu", "sigmoid", "tanh", "softplus"),
  normalization = c("none", "batch", "layer"),
 init_method = c("default", "xavier_uniform", "xavier_normal", "kaiming_uniform",
    "kaiming_normal"),
  epochs = 100L,
  lr = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.2,
  batch_size = 32L,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

`preprocess`	Optional preprocessing/normalization object.
`input_size`	Integer. Number of lagged inputs per training example.
`input_map`	Lag-selection strategy object, typically created by `tspredit::ts_lagmap()`.
`hidden_sizes`	Integer vector with hidden layer sizes.
`dropout`	Numeric. Dropout rate.
`activation`	Character. Hidden activation function. One of `"relu"`, `"leaky_relu"`, `"elu"`, `"gelu"`, or `"tanh"`.

output_activation
: Character. Output activation of the regression head. One of "none", "relu", "sigmoid", "tanh", or "softplus".

normalization
: Character. Optional normalization after each hidden linear layer. One of "none", "batch", or "layer".

init_method
: Character. Weight initialization strategy. One of "default", "xavier_uniform", "xavier_normal", "kaiming_uniform", or "kaiming_normal".

epochs
: Integer. Maximum number of training epochs. Default is 100L.

lr
: Numeric. Optimizer learning rate.

validation_strategy
: Character. One of static or dynamic.

stopping_rule
: Character. One of none, patience, sma, ema, or h.

val_ratio
: Numeric. Validation fraction used when validation is enabled.

batch_size
: Integer. Mini-batch size.

patience
: Integer. Early stopping patience.

min_delta
: Numeric. Minimum improvement to reset early stopping.

sma_window
: Integer. Window size used by sma.

ema_alpha
: Numeric. Smoothing factor used by ema.

test_window
: Integer. Window size used by h.

p_value
: Numeric. Significance threshold used by h.

Details

The object follows the tspredit::ts_regsw() contract: fit() receives supervised lag matrices and predict() returns a plain numeric vector, even when upstream time-series wrappers attach auxiliary metadata to forecast objects.

Value

A torch_ts_mlp object.

Examples

```
## Not run:
library(daltoolboxdp)
model <- torch_ts_mlp(
  input_size = 12,
  input_map = tspredit::ts_lagmap("pacf"),
  hidden_sizes = c(32L, 16L),
  normalization = "batch",
  init_method = "kaiming_uniform",
  epochs = 100L
)

## End(Not run)
```

ts_conv1d *Conv1D*

Description

Time-series forecaster using a configurable 1D convolutional neural network with unified training strategies and a Python/PyTorch backend.

Usage

```
ts_conv1d(
  preprocess = NA,
  input_size = NA,
  input_map = tspredit::ts_lagmap(),
  in_channels = 1L,
  sequence_length = NULL,
  conv_channels = 64L,
  kernel_sizes = NULL,
  strides = 1L,
  pooling = c("none", "max", "avg"),
  pool_kernel_size = 2L,
  dense_hidden_sizes = 50L,
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  epochs = 100L,
  lr = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.2,
  batch_size = 8L,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

preprocess	Optional preprocessing/normalization object.
input_size	Integer. Number of lagged inputs per training example.
input_map	Lag-selection strategy object, typically created by tspredit::ts_lagmap().
in_channels	Integer. Number of channels used to reshape each example before the convolution. input_size must equal in_channels * sequence_length.
sequence_length	Optional integer. Temporal length after reshaping. If NULL, it is inferred as input_size / in_channels.

`conv_channels` Integer vector. Output channels for each convolutional block.

`kernel_sizes` Integer vector. Kernel sizes for each convolutional block. If `NULL`, defaults to `2L` for sequence lengths greater than 1 and `1L` otherwise.

`strides` Integer vector. Strides for each convolutional block.

`pooling` Character. Pooling strategy applied after each convolutional block. One of `"none"`, `"max"`, or `"avg"`.

`pool_kernel_size` Integer. Pooling kernel size when pooling is enabled.

`dense_hidden_sizes` Integer vector. Hidden sizes of the dense head after the convolutional stack.

`activation` Character. Activation function used in convolutional and dense hidden layers. One of `"relu"`, `"leaky_relu"`, `"elu"`, `"gelu"`, or `"tanh"`.

`epochs` Integer. Maximum number of training epochs. Default is `100L`.

`lr` Numeric. Optimizer learning rate.

`validation_strategy` Character. One of `static` or `dynamic`.

`stopping_rule` Character. One of `none`, `patience`, `sma`, `ema`, or `h`.

`val_ratio` Numeric. Validation fraction used when validation is enabled.

`batch_size` Integer. Mini-batch size.

`patience` Integer. Early stopping patience.

`min_delta` Numeric. Minimum improvement to reset early stopping.

`sma_window` Integer. Window size used by `sma`.

`ema_alpha` Numeric. Smoothing factor used by `ema`.

`test_window` Integer. Window size used by `h`.

`p_value` Numeric. Significance threshold used by `h`.

Details

The Conv1D forecaster now supports multiple convolutional blocks, explicit channel/sequence reshaping, optional pooling, and a configurable dense prediction head. Keep the defaults to preserve the original single-channel behavior, or define architectures such as `conv_channels = c(32L, 64L)` and `dense_hidden_sizes = c(64L, 16L)`.

The object follows the `tspredit::ts_regsw()` contract: `fit()` receives supervised lag matrices and `predict()` returns a plain numeric vector, even when upstream time-series wrappers attach auxiliary metadata to forecast objects.

Value

A `ts_conv1d` object.

Examples

```
## Not run:
library(daltoolboxdp)
model <- ts_conv1d(
  input_size = 12,
  input_map = tspredit::ts_lagmap("acf"),
  in_channels = 1L,
  conv_channels = c(32L, 64L),
  dense_hidden_sizes = c(64L, 16L),
  epochs = 100L
)

## End(Not run)
```

`ts_lstm` *LSTM*

Description

Time-series forecaster using a configurable LSTM neural network with unified training strategies and a Python/PyTorch backend.

Usage

```
ts_lstm(
  preprocess = NA,
  input_size = NA,
  input_map = tspredit::ts_lagmap(),
  hidden_size = NULL,
  sequence_length = 1L,
  num_layers = 1L,
  dropout = 0,
  bidirectional = FALSE,
  mlp_hidden_sizes = integer(0),
  activation = c("relu", "leaky_relu", "elu", "gelu", "tanh"),
  epochs = 100L,
  lr = 0.001,
  validation_strategy = c("static", "dynamic"),
  stopping_rule = c("none", "patience", "sma", "ema", "h"),
  val_ratio = 0.2,
  batch_size = 8L,
  patience = 100L,
  min_delta = 1e-04,
  sma_window = 5L,
  ema_alpha = 0.2,
  test_window = 30L,
  p_value = 0.05
)
```

Arguments

preprocess	Optional preprocessing/normalization object.
input_size	Integer. Number of lagged inputs per training example.
input_map	Lag-selection strategy object, typically created by tspredit::ts_lagmap().
hidden_size	Optional integer. Hidden size used inside the LSTM. If NULL, defaults to input_size.
sequence_length	Integer. Number of time steps represented by each row. input_size must be divisible by sequence_length. Default is 1L.
num_layers	Integer. Number of LSTM layers.
dropout	Numeric. Recurrent dropout applied between LSTM layers when num_layers > 1.
bidirectional	Logical. Whether the LSTM is bidirectional.
mlp_hidden_sizes	Integer vector. Hidden sizes of the dense head applied after the LSTM output.
activation	Character. Activation function used in the dense head. One of "relu", "leaky_relu", "elu", "gelu", or "tanh".
epochs	Integer. Maximum number of training epochs. Default is 100L.
lr	Numeric. Optimizer learning rate.
validation_strategy	Character. One of static or dynamic.
stopping_rule	Character. One of none, patience, sma, ema, or h.
val_ratio	Numeric. Validation fraction used when validation is enabled.
batch_size	Integer. Mini-batch size.
patience	Integer. Early stopping patience.
min_delta	Numeric. Minimum improvement to reset early stopping.
sma_window	Integer. Window size used by sma.
ema_alpha	Numeric. Smoothing factor used by ema.
test_window	Integer. Window size used by h.
p_value	Numeric. Significance threshold used by h.

Details

The LSTM forecaster now supports multiple recurrent layers, dropout, bidirectionality, an optional dense head after the recurrent block, and explicit reshaping of each row into a sequence via sequence_length. Keeping sequence_length = 1L reproduces the previous behavior.

The object follows the tspredit::ts_regsw() contract: fit() receives supervised lag matrices and predict() returns a plain numeric vector, even when upstream time-series wrappers attach auxiliary metadata to forecast objects.

Value

A ts_lstm object.

Examples

```
## Not run:
library(daltoolboxdp)
model <- ts_lstm(
  input_size = 12,
  input_map = tspredit::ts_lagmap("seasonal", seasonality = 4),
  hidden_size = 16L,
  sequence_length = 3L,
  num_layers = 2L,
  dropout = 0.1,
  mlp_hidden_sizes = c(16L, 8L),
  epochs = 100L
)

## End(Not run)
```

Index