Package 'DeepLearningCausal'

October 30, 2025

Type Package

Title Causal Inference with Super Learner and Deep Neural Networks

Version 0.0.107

Maintainer Nguyen K. Huynh <khoinguyen.huynh@r.hit-u.ac.jp>

Description
Functions for deep learning estimation of Conditional Average Treatment Effects (CATEs) from meta-
learner models and Population Average Treatment Effects on the Treated (PATT) in settings with treatment noncompliance using reticulate, TensorFlow and Keras3. Functions in the package also implements
the conformal prediction framework that enables computation and illustration of conformal prediction (CP)
intervals for estimated individual treatment effects (ITEs) from meta-learner models. Additional functions in the package permit users to estimate the meta-
learner CATEs and the PATT in settings with
treatment noncompliance using weighted ensemble learning via the super learner approach and R neural networks.

License GPL-3

Encoding UTF-8

LazyData true

Imports ROCR, caret, neuralnet, SuperLearner, ggplot2, tidyr, magrittr, reticulate, keras3, Hmisc

Suggests testthat (>= 3.0.0), dplyr, class, xgboost, randomForest, glmnet, ranger, gam, e1071, gbm, tensorflow

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

URL https://github.com/hknd23/DeepLearningCausal

BugReports https://github.com/hknd23/DeepLearningCausal/issues

Config/testthat/edition 3

NeedsCompilation no

Author Nguyen K. Huynh [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6234-7232>),
Bumba Mukherjee [aut] (ORCID: <https://orcid.org/0000-0002-3453-601X>),
Yang Yang [aut] (ORCID: <https://orcid.org/0009-0004-6135-4555>)

Repository CRAN

Date/Publication 2025-10-30 10:30:02 UTC

Contents

complier_mod *Train complier model using ensemble methods*

Description

Train model using group exposed to treatment with compliance as binary outcome variable and covariates.

Usage

```
complier_mod(
  exp.data,
  complier.formula,
  treat.var,
  ID = NULL,
  SL.learners = c("SL.glmnet", "SL.xgboost", "SL.ranger", "SL.nnet", "SL.glm")
)
```

Arguments

exp.data	list object of experimental data.
complier.formula	formula to fit compliance model (c ~ x) using complier variable and covariates
treat.var	string specifying the binary treatment variable
ID	string for name of identifier variable.
SL.learners	vector of strings for ML classifier algorithms. Defaults to extreme gradient boosting, elastic net regression, random forest, and neural nets.

Value

model object of trained model.

complier_predict *Complier model prediction*

Description

Predict Compliance from control group in experimental data

Usage

```
complier_predict(complier.mod, exp.data, treat.var, compl.var)
```

Arguments

complier.mod	output from trained ensemble superlearner model
exp.data	data.frame object of experimental dataset
treat.var	string specifying the binary treatment variable
compl.var	string specifying binary complier variable

Value

data.frame object with true compliers, predicted compliers in the control group, and all compliers (actual + predicted).

---

conformal_plot *conformal_plot*

Description

Visualizes the distribution of estimated individual treatment effects (ITEs) along with their corresponding conformal prediction intervals. The function randomly samples a proportion of observations from a fitted metalearner_ensemble or metalearner_deeplearning object and plots the conformal intervals as vertical ranges around the point estimates. This allows users to visually assess the uncertainty and variation in estimated treatment effects.

Usage

```
conformal_plot(
  x,
  ...,
  seed = 1234,
  prop = 0.3,
  binary.outcome = FALSE,
  x.labels = TRUE,
  x.title = "Observations",
  color = "steelblue",
  break.by = 0.5
)
```

Arguments

x	A fitted model object of class metalearner_ensemble or metalearner_deeplearning that contains a conformal_interval element.
...	Additional arguments (currently unused).
seed	Random seed for reproductibility of subsampling. Default is 1234.
prop	Proportion of observations to randomly sample for plotting. Must be between 0 and 1. Default is 0.3.

binary.outcome	Logical; if TRUE, constrains the y-axis to [-1, 1] for binary outcomes. Default is FALSE.
x.labels	Logical; if TRUE, displays x-axis labels for each sampled observation. Default is TRUE.
x.title	Character string specifying the x-axis title. Default is "Observations".
color	Color of the conformal intervals and points. Default is "steelblue".
break.by	Numeric value determining the spacing between y-axis breaks. Default is 0.5.

Details

The function extracts the estimated ITEs (CATEs) and conformal intervals (ITE_lower, ITE_upper) from the model output, samples a subset of rows, and generates a ggplot2 visualization. Each vertical line represents the conformal prediction interval for one observation's treatment effect estimate. The conformal intervals are typically obtained from weighted split-conformal inference, using propensity overlap weights to adjust interval width.

Value

A ggplot object showing sampled individual treatment effects with their weighted conformal prediction intervals.

---

deep_complier_mod *Train complier model using deep neural learning through Tensorflow*

Description

Train model using group exposed to treatment with compliance as binary outcome variable and covariates.

Usage

```
deep_complier_mod(
  complier.formula,
  exp.data,
  treat.var,
  algorithm = "adam",
  hidden.layer = c(2, 2),
  hidden_activation = "relu",
  ID = NULL,
  epoch = 10,
  verbose = 1,
  batch_size = 32,
  validation_split = NULL,
  patience = NULL,
  dropout_rate = NULL
)
```

Arguments

complier.formula	formula to fit compliance model (c ~ x) using complier variable and covariates
exp.data	list object of experimental data.
treat.var	string specifying the binary treatment variable
algorithm	string for name of optimizer algorithm. Set to adam. other optimization algorithms available are sgd, rprop, adagrad.
hidden.layer	vector specifying the hidden layers and the number of neurons in each layer.
hidden_activation	string or vector for activation function used for hidden layers. Defaults to "relu".
ID	string for name of identifier variable.
epoch	integer for number of epochs
verbose	1 to display model training information and learning curve plot. 0 to suppress messages and plots.
batch_size	integer for batch size to split the training set. Defaults to 32.
validation_split	double for proportion of training data to be split for validation.
patience	integer for number of epochs with no improvement after which training will be stopped.
dropout_rate	double or vector for proportion of hidden layer to drop out.

Value

deep.complier.mod model object

---

deep_predict *Complier model prediction*

Description

Predict Compliance from control group in experimental data

Usage

```
deep_predict(
  deep.complier.mod,
  complier.formula,
  exp.data,
  treat.var,
  compl.var
)
```

Arguments

deep.complier.mod	model object from deep.complier.mod()
complier.formula	formula to fit compliance model (c ~ x) using complier variable and covariates
exp.data	data.frame object of experimental dataset
treat.var	string specifying the binary treatment variable
compl.var	string specifying binary complier variable

Value

data.frame object with true compliers, predicted compliers in the control group, and all compliers (actual + predicted).

---

deep_response_model *Response model from experimental data using deep neural learning through Tensorflow*

Description

Train response model (response variable as outcome and covariates) from all compliers (actual + predicted) in experimental data using Tensorflow.

Usage

```
deep_response_model(
  response.formula,
  exp.data,
  exp.compliers,
  compl.var,
  algorithm = "adam",
  hidden.layer = c(2, 2),
  hidden_activation = "relu",
  epoch = 10,
  verbose = 1,
  batch_size = 32,
  output_units = 1,
  validation_split = NULL,
  patience = NULL,
  output_activation = "linear",
  loss = "mean_squared_error",
  metrics = "mean_squared_error",
  dropout_rate = NULL
)
```

Arguments

response.formula	formula specifying the response variable and covariates.
exp.data	experimental dataset.
exp.compliers	data.frame object of compliers from complier_predict.
compl.var	string specifying binary complier variable
algorithm	string for optimizer algorithm in response model.
hidden.layer	vector specifying hidden layers and the number of neurons in each hidden layer
hidden_activation	string or vector for activation functions in hidden layers.
epoch	integer for number of epochs
verbose	1 to display model training information and learning curve plot. 0 to suppress messages and plots.
batch_size	batch size to split training data.
output_units	integer for units in output layer. Defaults to 1 for continuous and binary outcome variables. In case of multinomial outcome variable, value should be set to the number of categories.
validation_split	double for the proportion of test data to be split as validation in response model.
patience	integer for number of epochs with no improvement after which training will be stopped.
output_activation	string for activation function in output layer. "linear" is recommended for continuous outcome variables, and "sigmoid" for binary outcome variables
loss	string for loss function. "mean_squared_error" recommended for linear models, "binary_crossentropy" for binary models.
metrics	string for metrics. "mean_squared_error" recommended for linear models, "binary_accuracy" for binary models.
dropout_rate	double or vector for proportion of hidden layer to drop out in response model.

Value

model object of trained response model.

---

exp_data *Survey Experiment of Support for Populist Policy*

Description

Shortened version of survey response data that incorporates a vignette survey experiment. The vignette describes an international crisis between country A and B. After reading this vignette, respondents are randomly assigned to the control group or to one of two treatments: policy prescription to said crisis by strong (populist) leader and centrist (non-populist) leader. The respondents are then asked whether they are willing to support the policy decision to fight a war against country A, which is the dependent variable.

Usage

```
data(exp_data)
```

Format

exp_data:
A data frame with 257 rows and 12 columns:

female Gender.
age Age of participant.
income Monthly household income.
religion Religious denomination
practicing_religion Importance of religion in life.
education Educational level of participant.
political_ideology Political ideology of participant.
employment Employment status of participant.
marital_status Marital status of participant.
job_loss Concern about job loss.
strong_leader Binary treatment measure of leader type.
support_war Binary outcome measure for willingness to fight war. #' ...

Source

Yadav and Mukherjee (2024)

---

exp_data_full *Survey Experiment of Support for Populist Policy*

Description

Extended experiment data with 514 observations

Usage

```
data(exp_data_full)
```

Format

exp_data_full:
A data frame with 514 rows and 12 columns:

female Gender.
age Age of participant.
income Monthly household income.
religion Religious denomination
practicing_religion Importance of religion in life.

education Educational level of participant.
political_ideology Political ideology of participant.
employment Employment status of participant.
marital_status Marital status of participant.
job_loss Concern about job loss.
strong_leader Binary treatment measure of leader type.
support_war Binary outcome measure for willingness to fight war. #' ...

Source

Yadav and Mukherjee (2024)

`hte_plot` *hte_plot*

Description

Produces plot to illustrate sub-group Heterogeneous Treatment Effects (HTE) of estimated CATEs from `metalearner_ensemble` and `metalearner_neural`, as well as PATT-C from `pattc_ensemble` and `pattc_neural`.

Usage

```
hte_plot(
  x,
  ...,
  boot = TRUE,
  n_boot = 1000,
  cut_points = NULL,
  custom_labels = NULL,
  zero_int = TRUE,
  selected_vars = NULL
)
```

Arguments

`x`	estimated model from `metalearner_ensemble`, `metalearner_neural`, `pattc_ensemble`, or `pattc_neural`.
`...`	Additional arguments
`boot`	logical for using bootstraps to estimate confidence intervals.
`n_boot`	number of bootstrap iterations. Only used with boot = TRUE.
`cut_points`	numeric vector for cut-off points to generate subgroups from covariates. If left blank a vector generated from median values will be used.
`custom_labels`	character vector for the names of subgroups.
`zero_int`	logical for vertical line at 0 x intercept.
`selected_vars`	vector for names of covariates to use for subgroups.

Value

ggplot object illustrating subgroup HTE and 95% confidence intervals.

Examples

```
# load dataset
set.seed(123456)
xlearner_nn <- metalearner_neural(cov.formula = support_war ~ age +
                                  income  + employed  + job_loss,
                                  data = exp_data,
                                  treat.var = "strong_leader",
                                  meta.learner.type = "X.Learner",
                                  stepmax = 2e+9,
                                  nfolds = 5,
                                  algorithm = "rprop+",
                                  hidden.layer = c(3),
                                  linear.output = FALSE,
                                  binary.preds = FALSE)

hte_plot(xlearner_nn)
hte_plot(xlearner_nn,
        selected_vars = c("age", "income"),
        cut_points = c(33, 3),
        custom_labels = c("Age <= 33", "Age > 33", "Income <= 3", "Income > 3"),
        n_boot = 500)
```

`metalearner_deeplearning` *metalearner_deeplearning*

Description

`metalearner_deeplearning` implements the meta learners for estimating CATEs using Deep Neural Networks through Tensorflow. Deep Learning Estimation of CATEs from four meta-learner models (S,T,X and R-learner) using TensorFlow and Keras3

Usage

```
metalearner_deeplearning(
  data = NULL,
  train.data = NULL,
  test.data = NULL,
  cov.formula,
  treat.var,
  meta.learner.type,
  nfolds = 5,
  algorithm = "adam",
```

```
  hidden.layer = c(2, 2),
  hidden_activation = "relu",
  output_activation = "linear",
  output_units = 1,
  loss = "mean_squared_error",
  metrics = "mean_squared_error",
  epoch = 10,
  verbose = 1,
  batch_size = 32,
  validation_split = NULL,
  patience = NULL,
  dropout_rate = NULL,
  conformal = FALSE,
  alpha = 0.1,
  calib_frac = 0.5,
  prob_bound = TRUE,
  seed = 1234
)
```

Arguments

`data`	data.frame object of data. If a single dataset is specified, then the model will use cross-validation to train the meta-learners and estimate CATEs. Users can also specify the arguments (defined below) to separately train meta-learners on their training data and estimate CATEs with their test data.
`train.data`	data.frame object of training data for Train/Test mode. Argument must be specified to separately train the meta-learners on the training data.
`test.data`	data.frame object of test data for Train/Test mode. Argument must be specified to estimate CATEs on the test data.
`cov.formula`	formula description of the model y ~ x(list of covariates). Permits users to specify covariates in the meta-learner model of interest. This includes the outcome variables and the confounders.
`treat.var`	string for name of Treatment variable. Users can specify the treatment variable in their data by employing the treat.var argument.
`meta.learner.type`	string of "S.Learner", "T.Learner", "X.Learner", or "R.Learner". Employed to specify any of the following four meta-learner models for estimation via deep learning: S,T,X or R-Learner.
`nfolds`	integer for number of folds for Meta Learners. When a single dataset is specified, then users employ cross-validation to train the meta-learners and estimate CATEs. For a single dataset, users specify nfolds to define the number of folds to split data for cross-validation.
`algorithm`	string for optimization algorithm. For optimizers available see `keras` package. Arguments to reconfigure and train the deep neural networks for meta-learner estimation include the optimization algorithm. Options for the optimization alogrithm include "adam", “adagrad”, “rmsprop”, “sgd”.

`hidden.layer` permits users to specify the number of hidden layers in the model and the number of neurons in each hidden layer.

`hidden_activation` string or vector for name of activation function for hidden layers of model. Defaults to "relu" which means that users can specify a single value to use one activation function for each hidden layer. While "relu" is a popular choice for hidden layers, users can also use "softmax" which converts a vector of values into a probability distribution and "tanh" that maps input to a value between -1 and 1.

`output_activation` string for name of activation function for output layer of model. "linear" is recommended for continuous outcome variables, and "sigmoid" for binary outcome variables. For activation functions available see `keras` package. 'For instance, Keras provides various activation functions that can be used in neural network layers to introduce non-linearity

`output_units` integer for units in output layer. Defaults to 1 for continuous and binary outcome variables. In case of multinomial outcome variable, set to the number of categories.

`loss` string for loss function "mean_squared_error" recommended for linear models, "binary_crossentropy" for binary models.

`metrics` string for metrics in response model. "mean_squared_error" recommended for linear models, "binary_accuracy" for binary models.

`epoch` interger for number of epochs. epoch denotes one complete pass through the entire training dataset. Model processes each training example once during an epoch.

`verbose` integer specifying the verbosity level during training. 1 for full information and learning curve plots. 0 to suppress messages and plots.

`batch_size` integer for batch size to split training data. batch size refers to the number of training samples processed before the model's parameters are updated. Batch size is a vital hyperparameter that affects both training speed and model performance. It is crucial for computational efficiency.

`validation_split` double for proportion of training data to split for validation. validation split involves partitioning data into training and validation sets to build and tune model.

`patience` integer for number of epochs with no improvement to wait before stopping training. patience stops training of neural network if model's performance on validation data stops improving.

`dropout_rate` double or vector for proportion of hidden layer to drop out. dropout rate is hyperparameter for preventing a model from overfitting the training data.

`conformal` logical for whether to compute conformal prediction intervals conformal prediction intervals provide measure of uncertainty for ITEs.

`alpha` proportion for conformal prediction intervals alpha proportion refers to significance level that guarantees desired coverage probability for ITEs

`calib_frac` fraction of training data to use for calibration in conformal inference

prob_bound	logical for whether to bound conformal intervals within [-1,1] for classification models
seed	random seed

Value

metalearner_deeplearning object with CATEs

Examples

```
## Not run:
#check for python and required modules
python_ready()
data("exp_data")

s_deeplearning <- metalearner_deeplearning(data = exp_data,
cov.formula  = support_war ~ age + female + income + education
+ employed + married + hindu + job_loss,
treat.var = "strong_leader",  meta.learner.type = "S.Learner",
nfolds = 5,  algorithm = "adam",
hidden.layer = c(2,2),  hidden_activation = "relu",
output_activation = "sigmoid", output_units = 1,
loss = "binary_crossentropy",  metrics = "accuracy",
epoch = 10,  verbose = 1,   batch_size = 32,
validation_split = NULL,  patience = NULL,
dropout_rate = NULL, conformal= FALSE,  seed=1234)

## End(Not run)
## Not run:
#check for python and required modules
python_ready()
data("exp_data")

t_deeplearning <- metalearner_deeplearning(data = exp_data,
cov.formula  = support_war ~ age + female + income + education
+ employed + married + hindu + job_loss,
treat.var = "strong_leader",  meta.learner.type = "T.Learner",
nfolds = 5,  algorithm = "adam",
hidden.layer = c(2,2),  hidden_activation = "relu",
output_activation = "sigmoid", output_units = 1,
loss = "binary_crossentropy",  metrics = "accuracy",
epoch = 10,  verbose = 1,   batch_size = 32,
validation_split = NULL,  patience = NULL,
dropout_rate = NULL, conformal= TRUE,
alpha = 0.1,calib_frac = 0.5, prob_bound = TRUE, seed = 1234)

## End(Not run)
```

`metalearner_ensemble` *metalearner_ensemble*

Description

`metalearner_ensemble` implements the S-learner, T-learner, and X-learner for weighted ensemble learning estimation of CATEs using super learner. The super learner in this case includes the following machine learning algorithms: extreme gradient boosting, glmnet (elastic net regression), random forest and neural nets.

Usage

```
metalearner_ensemble(
  data = NULL,
  train.data = NULL,
  test.data = NULL,
  cov.formula,
  treat.var,
  meta.learner.type,
  SL.learners = c("SL.glmnet", "SL.xgboost", "SL.nnet"),
  nfolds = 5,
  family = gaussian(),
  binary.preds = FALSE,
  conformal = FALSE,
  alpha = 0.1,
  calib_frac = 0.5,
  seed = 1234
)
```

Arguments

`data`	`data.frame` object of data for cross-validation
`train.data`	`data.frame` object of training data argument to separately train the meta-learners on training data.
`test.data`	`data.frame` object of test data argument to estimate CATEs on the test data.
`cov.formula`	formula description of the model y ~ x(list of covariates) permits users to incorporate outcome variable and confounders in model.
`treat.var`	string for the name of treatment variable.
`meta.learner.type`	string specifying is the S-learner and `"T.Learner"` for the T-learner model. `"X.Learner"` for the X-learner model. `"R.Learner"` for the X-learner model.
`SL.learners`	vector for super learner ensemble that includes extreme gradient boosting, glmnet, random forest, and neural nets.
`nfolds`	number of folds for cross-validation. Currently supports up to 5 folds.
`family`	gaussian() or binomial() family for outcome variable. 5 folds.

binary.preds	logical for whether outcome predictions should be binary
conformal	logical for whether to compute conformal prediction intervals
alpha	proportion for conformal prediction intervals
calib_frac	fraction of training data to use for calibration in conformal inference
seed	random seed

Value

metalearner_ensemble of predicted outcome values and CATEs estimated by the meta learners for each observation.

Examples

```
# load dataset
data(exp_data)
#load SuperLearner package
library(SuperLearner)
# estimate CATEs with S Learner
set.seed(123456)
slearner <- metalearner_ensemble(cov.formula = support_war ~ age +
                                   income + employed + job_loss,
                                 data = exp_data,
                                 treat.var = "strong_leader",
                                 meta.learner.type = "S.Learner",
                                 SL.learners = c("SL.glm"),
                                 nfolds = 5,
                                 binary.preds = FALSE,
                                 )
print(slearner)


# estimate CATEs with T Learner
set.seed(123456)
tlearner <- metalearner_ensemble(cov.formula = support_war ~ age + income +
                                   employed  + job_loss,
                                   data = exp_data,
                                   treat.var = "strong_leader",
                                   meta.learner.type = "T.Learner",
                                   SL.learners = c("SL.xgboost",
                                               "SL.nnet"),
                                   nfolds = 5,
                                   binary.preds = FALSE,
                                   )

print(tlearner)



# estimate CATEs with X Learner
set.seed(123456)
xlearner <- metalearner_ensemble(cov.formula = support_war ~ age + income +
```

```
 employed  + job_loss,
                                test.data = exp_data,
                                train.data = exp_data,
                                treat.var = "strong_leader",
                                meta.learner.type = "X.Learner",
                                SL.learners = c("SL.glmnet","SL.xgboost",
                                "SL.nnet"),
                                binary.preds = TRUE)

print(xlearner)
```

`metalearner_neural` *metalearner_neural*

Description

`metalearner_neural` implements the S-learner and T-learner for estimating CATE using Deep Neural Networks. The Resilient back propagation (Rprop) algorithm is used for training neural networks.

Usage

```
metalearner_neural(
  data,
  cov.formula,
  treat.var,
  meta.learner.type,
  stepmax = 1e+05,
  nfolds = 5,
  algorithm = "rprop+",
  hidden.layer = c(4, 2),
  act.fct = "logistic",
  err.fct = "sse",
  linear.output = TRUE,
  binary.preds = FALSE
)
```

Arguments

`data`	`data.frame` object of data.
`cov.formula`	formula description of the model y ~ x(list of covariates).
`treat.var`	string for the name of treatment variable.
`meta.learner.type`	string specifying is the S-learner and `"T.Learner"` for the T-learner model. `"X.Learner"` for the X-learner model. `"R.Learner"` for the R-learner model.

stepmax	maximum number of steps for training model.
nfolds	number of folds for cross-validation. Currently supports up to 5 folds.
algorithm	a string for the algorithm for the neural network. Default set to rprop+, the Resilient back propagation (Rprop) with weight backtracking algorithm for training neural networks.
hidden.layer	vector of integers specifying layers and number of neurons.
act.fct	"logistic" or "tanh" for activation function to be used in the neural network.
err.fct	"ce" for cross-entropy or "sse" for sum of squared errors as error function.
linear.output	logical specifying regression (TRUE) or classification (FALSE) model.
binary.preds	logical specifying predicted outcome variable will take binary values or proportions.

Value

metalearner_neural of predicted outcome values and CATEs estimated by the meta learners for each observation.

Examples

```
# load dataset
data(exp_data)
# estimate CATEs with S Learner
set.seed(123456)
slearner_nn <- metalearner_neural(cov.formula = support_war ~ age + income +
                                  employed  + job_loss,
                                  data = exp_data,
                                  treat.var = "strong_leader",
                                  meta.learner.type = "S.Learner",
                                  stepmax = 2e+9,
                                  nfolds = 5,
                                  algorithm = "rprop+",
                                  hidden.layer = c(1),
                                  linear.output = FALSE,
                                  binary.preds = FALSE)

print(slearner_nn)


# load dataset
set.seed(123456)
# estimate CATEs with T Learner
tlearner_nn <- metalearner_neural(cov.formula = support_war ~ age +
                                  income  +
                                  employed  + job_loss,
                                  data = exp_data,
                                  treat.var = "strong_leader",
                                  meta.learner.type = "T.Learner",
                                  stepmax = 1e+9,
                                  nfolds = 5,
```

```
                                    algorithm = "rprop+",
                                    hidden.layer = c(2,1),
                                    linear.output = FALSE,
                                    binary.preds = FALSE)

print(tlearner_nn)



# load dataset
set.seed(123456)
# estimate CATEs with X Learner
xlearner_nn <- metalearner_neural(cov.formula = support_war ~ age +
                                    income  +
                                    employed  + job_loss,
                                    data = exp_data,
                                    treat.var = "strong_leader",
                                    meta.learner.type = "X.Learner",
                                    stepmax = 2e+9,
                                    nfolds = 5,
                                    algorithm = "rprop+",
                                    hidden.layer = c(3),
                                    linear.output = FALSE,
                                    binary.preds = FALSE)

print(xlearner_nn)
```

`neuralnet_complier_mod` *Train compliance model using neural networks*

Description

Train model using group exposed to treatment with compliance as binary outcome variable and covariates.

Usage

```
neuralnet_complier_mod(
  complier.formula,
  exp.data,
  treat.var,
  algorithm = "rprop+",
  hidden.layer = c(4, 2),
  act.fct = "logistic",
  ID = NULL,
  stepmax = 1e+08
)
```

Arguments

complier.formula	formula for complier variable as outcome and covariates (c ~ x)
exp.data	data.frame for experimental data.
treat.var	string for treatment variable.
algorithm	string for algorithm for training neural networks. Default set to the Resilient back propagation with weight backtracking (rprop+). Other algorithms include backprop', rprop-', 'sag', or 'slr' (see neuralnet package).
hidden.layer	vector for specifying hidden layers and number of neurons.
act.fct	"logistic" or "tanh activation function.
ID	string for identifier variable
stepmax	maximum number of steps.

Value

trained complier model object

---

neuralnet_pattc_counterfactuals

Assess Population Data counterfactuals

Description

Create counterfactual datasets in the population for compliers and noncompliers. Then predict potential outcomes using trained model from neuralnet_response_model.

Usage

```
neuralnet_pattc_counterfactuals(
  pop.data,
  neuralnet.response.mod,
  ID = NULL,
  cluster = NULL,
  binary.preds = FALSE
)
```

Arguments

pop.data	population data.
neuralnet.response.mod	trained model from. neuralnet_response_model.
ID	string for identifier variable.
cluster	string for clustering variable (currently unused).
binary.preds	logical specifying predicted outcome variable will take binary values or proportions.

Value

data.frame of predicted outcomes of response variable from counterfactuals.

---

neuralnet_predict *Predicting Compliance from experimental data*

Description

Predicting Compliance from control group experimental data

Usage

```
neuralnet_predict(neuralnet.complier.mod, exp.data, treat.var, compl.var)
```

Arguments

neuralnet.complier.mod	results from neuralnet_complier_mod
exp.data	data.frame of experimental data
treat.var	string for treatment variable
compl.var	string for compliance variable

Value

data.frame object with true compliers, predicted compliers in the control group, and all compliers (actual + predicted).

---

neuralnet_response_model *Modeling Responses from experimental data Using Deep NN*

Description

Model Responses from all compliers (actual + predicted) in experimental data using neural network.

Usage

```
neuralnet_response_model(
  response.formula,
  exp.data,
  neuralnet.compliers,
  compl.var,
  algorithm = "rprop+",
  hidden.layer = c(4, 2),
  act.fct = "logistic",
  err.fct = "sse",
  linear.output = TRUE,
  stepmax = 1e+08
)
```

Arguments

response.formula	formula for response variable and covariates (y ~ x)
exp.data	data.frame of experimental data.
neuralnet.compliers	data.frame of compliers (actual + predicted) from neuralnet_predict.
compl.var	string of compliance variable
algorithm	neural network algorithm, default set to "rprop+".
hidden.layer	vector specifying hidden layers and number of neurons.
act.fct	"logistic" or "tanh activation function.
err.fct	"sse" for sum of squared errors or "ce" for cross-entropy.
linear.output	logical for whether output (outcome variable) is linear or not.
stepmax	maximum number of steps for training model.

Value

trained response model object

---

pattc_counterfactuals *Assess Population Data counterfactuals*

Description

Create counterfactual datasets in the population for compliers and noncompliers. Then predict potential outcomes from counterfactuals.

Usage

```
pattc_counterfactuals(
  pop.data,
  response.mod,
  ID = NULL,
  cluster = NULL,
  binary.preds = FALSE
)
```

Arguments

pop.data	population dataset
response.mod	trained model from response_model.
ID	string fir identifier variable
cluster	string for clustering variable
binary.preds	logical specifying whether predicted outcomes are proportions or binary (0-1).

Value

data.frame object of predicted outcomes of counterfactual groups.

---

pattc_deeplearning *Deep PATT-C*

Description

This function implements the Deep PATT-C method for estimating the Population Average Treatment Effect on the Treated Compliers (PATT-C) using deep learning models using keras and Tensorflow. It consists of training a deep learning model to predict compliance among treated individuals, predicting compliance in the experimental data, training a response model among predicted compliers, and estimating counterfactual outcomes in the population data.

Usage

```
pattc_deeplearning(
  response.formula,
  compl.var,
  treat.var,
  exp.data,
  pop.data,
  compl.algorithm = "adam",
  response.algorithm = "adam",
  compl.hidden.layer = c(4, 2),
  response.hidden.layer = c(4, 2),
  compl.hidden_activation = "relu",
  response.hidden_activation = "relu",
```

```
  response.output_activation = "linear",
  response.output_units = 1,
  response.loss = "mean_squared_error",
  response.metrics = "mean_absolute_error",
  ID = NULL,
  weights = NULL,
  cluster = NULL,
  compl.epoch = 10,
  response.epoch = 10,
  compl.validation_split = NULL,
  response.validation_split = NULL,
  compl.patience = NULL,
  response.patience = NULL,
  compl.dropout_rate = NULL,
  response.dropout_rate = NULL,
  verbose = 1,
  batch_size = 32,
  nboot = 1000,
  seed = 1234
)
```

Arguments

response.formula	formula specifying the response variable and covariates.
compl.var	string specifying the name of the compliance variable.
treat.var	string specifying the name of the treatment variable.
exp.data	data frame containing the experimental data.
pop.data	data frame containing the population data.
compl.algorithm	string for name of optimizer algorithm for complier model. For optimizers available see keras package.
response.algorithm	string for name of optimizer algorithm for response model. For optimizers available see keras package.
compl.hidden.layer	vector specifying the hidden layers in the complier model and the number of neurons in each hidden layer.
response.hidden.layer	vector specifying the hidden layers in the response model and the number of neurons in each hidden layer.
compl.hidden_activation	string or vector for name of activation function for hidden layers complier model. Defaults to "relu" (Rectified Linear Unit)
response.hidden_activation	string or vector for name of activation function for hidden layers complier model. Defaults to "relu" (Rectified Linear Unit)

`response.output_activation`
: string for name of activation function for output layer of response model. "linear" is recommended for continuous outcome variables, and "sigmoid" for binary outcome variables. For activation functions available see `keras` package.

`response.output_units`
: integer for units in output layer. Defaults to 1 for continuous and binary outcome variables. In case of multinomial outcome variable, set to the number of categories.

`response.loss`
: string for loss function in response model. "mean_squared_error" recommended for linear models, "binary_crossentropy" for binary models.

`response.metrics`
: string for metrics in response model. "mean_squared_error" recommended for linear models, "binary_accuracy" for binary models.

`ID`
: optional string specifying the name of the identifier variable.

`weights`
: optional string specifying the name of the weights variable.

`cluster`
: optional string specifying the name of the clustering variable.

`compl.epoch`
: Integer for the number of epochs for complier model.

`response.epoch`
: integer for the number of epochs for response model.

`compl.validation_split`
: double for the proportion of test data to be split as validation in complier model. Defaults to 0.2.

`response.validation_split`
: double for the proportion of test data to be split as validation in response model. Defaults to 0.2.

`compl.patience`
: integer for number of epochs with no improvement after which training will be stopped in complier model.

`response.patience`
: integer for number of epochs with no improvement after which training will be stopped in response model.

`compl.dropout_rate`
: double or vector for proportion of hidden layer to drop out in complier model.

`response.dropout_rate`
: double or vector for proportion of hidden layer to drop out in response model.

`verbose`
: integer specifying the verbosity level during training. Defaults to 1.

`batch_size`
: integer specifying the batch size for training the deep learning models. Default is 32.

`nboot`
: integer specifying the number of bootstrap samples if bootstrap is TRUE. Default is 1000.

`seed`
: random seed

Value

pattc_deeplearning object containing the fitted models, predictions, counterfactuals, and PATT-C estimate.

Examples

```
## Not run:
#check for python and required modules
python_ready()

data("exp_data")
data("pop_data")
set.seed(1243)
deeppattc <- pattc_deeplearning(response.formula = support_war ~ age + female +
income + education +  employed + married + hindu + job_loss,
exp.data = exp_data, pop.data = pop_data,
treat.var = "strong_leader",  compl.var = "compliance",
compl.algorithm = "adam", response.algorithm = "adam",
compl.hidden.layer = c(4,2),  response.hidden.layer = c(4,2),
compl.hidden_activation = "relu",  response.hidden_activation = "relu",
response.output_activation = "sigmoid", response.output_units = 1,
response.loss = "binary_crossentropy", response.metrics = "accuracy",
compl.epoch = 50, response.epoch = 80,
verbose = 1, batch_size = 32,
compl.validation_split = 0.2, response.validation_split = 0.2,
compl.dropout_rate = 0.1, response.dropout_rate = 0.1,
compl.patience = 20, response.patience = 20,
nboot = 1000, seed = 1234)

## End(Not run)
```

---

pattc_deeplearning_counterfactuals

Assess Population Data counterfactuals

Description

Create counterfactual datasets in the population for compliers and noncompliers.

Usage

```
pattc_deeplearning_counterfactuals(
  pop.data,
  response.mod,
  response.formula,
  ID = NULL,
  cluster = NULL
)
```

Arguments

pop.data	population dataset
response.mod	trained model from response_model.

response.formula
: formula specifying the response variable and covariates.

ID
: string fir identifier variable

cluster
: string for clustering variable

Value

data.frame object of predicted outcomes of counterfactual groups.

---

pattc_ensemble *PATT-C SL Ensemble*

Description

pattc_ensemble estimates the Population Average Treatment Effect of the Treated from experimental data with noncompliers using the super learner ensemble that includes extreme gradient boosting, glmnet (elastic net regression), random forest and neural nets.

Usage

```
pattc_ensemble(
  response.formula,
  exp.data,
  pop.data,
  treat.var,
  compl.var,
 compl.SL.learners = c("SL.glmnet", "SL.xgboost", "SL.ranger", "SL.nnet", "SL.glm"),
 response.SL.learners = c("SL.glmnet", "SL.xgboost", "SL.ranger", "SL.nnet", "SL.glm"),
  response.family = gaussian(),
  ID = NULL,
  cluster = NULL,
  binary.preds = FALSE,
  bootstrap = FALSE,
  nboot = 1000
)
```

Arguments

response.formula
: formula for the effects of covariates on outcome variable (y ~ x).

exp.data
: data.frame object for experimental data. Must include binary treatment and compliance variable.

pop.data
: data.frame object for population data. Must include binary compliance variable.

treat.var
: string for binary treatment variable.

compl.var
: string for binary compliance variable.

`compl.SL.learners`
: vector of names of ML algorithms used for compliance model.

`response.SL.learners`
: vector of names of ML algorithms used for response model.

`response.family`
: gaussian() or binomial() for response model.

`ID`
: string for name of identifier. (currently not used)

`cluster`
: string for name of cluster variable. (currently not used)

`binary.preds`
: logical specifying predicted outcome variable will take binary values or proportions.

`bootstrap`
: logical for bootstrapped PATT-C.

`nboot`
: number of bootstrapped samples. Only used with `bootstrap = FALSE`

Value

`pattc_ensemble` object of results of t test as PATTC estimate.

Examples

```
# load datasets
data(exp_data_full) # full experimental data
data(exp_data) #experimental data
data(pop_data) #population data
#attach SuperLearner (model will not recognize learner if package is not loaded)
library(SuperLearner)
set.seed(123456)
#specify models and estimate PATTC

pattc_boot <- pattc_ensemble(response.formula = support_war ~ age + income +
                               education + employed + job_loss,
                               exp.data = exp_data_full,
                               pop.data = pop_data,
                               treat.var = "strong_leader",
                               compl.var = "compliance",
                               compl.SL.learners = c("SL.glm", "SL.nnet"),
                               response.SL.learners = c("SL.glm", "SL.nnet"),
                               response.family = binomial(),
                               ID = NULL,
                               cluster = NULL,
                               binary.preds = FALSE,
                               bootstrap = TRUE,
                               nboot = 1000)
print(pattc_boot)
```

pattc_neural *Estimate PATT_C using Deep NN*

Description

estimates the Population Average Treatment Effect of the Treated from experimental data with noncompliers using Deep Neural Networks.

Usage

```
pattc_neural(
  response.formula,
  exp.data,
  pop.data,
  treat.var,
  compl.var,
  compl.algorithm = "rprop+",
  response.algorithm = "rprop+",
  compl.hidden.layer = c(4, 2),
  response.hidden.layer = c(4, 2),
  compl.act.fct = "logistic",
  response.err.fct = "sse",
  response.act.fct = "logistic",
  linear.output = TRUE,
  compl.stepmax = 1e+08,
  response.stepmax = 1e+08,
  ID = NULL,
  cluster = NULL,
  binary.preds = FALSE,
  bootstrap = FALSE,
  nboot = 1000
)
```

Arguments

response.formula	formula of response variable as outcome and covariates (y ~ x)
exp.data	data.frame of experimental data. Must include binary treatment and compliance variables.
pop.data	data.frame of population data. Must include binary compliance variable
treat.var	string for treatment variable.
compl.var	string for compliance variable
compl.algorithm	string for algorithm to train neural network for compliance model. Default set to "rprop+". See (neuralnet package for available algorithms).

`response.algorithm`
: string for algorithm to train neural network for response model. Default set to `"rprop+"`. See (`neuralnet` package for available algorithms).

`compl.hidden.layer`
: vector for specifying hidden layers and number of neurons in complier model.

`response.hidden.layer`
: vector for specifying hidden layers and number of neurons in response model.

`compl.act.fct`
: "logistic" or "tanh" activation function for complier model.

`response.err.fct`
: "sse" for sum of squared errors or "ce" for cross-entropy for response model.

`response.act.fct`
: "logistic" or "tanh" activation function for response model.

`linear.output`
: logical for whether output (outcome variable) is linear or not for response model.

`compl.stepmax`
: maximum number of steps for complier model

`response.stepmax`
: maximum number of steps for response model

`ID`
: string for identifier variable

`cluster`
: string for cluster variable.

`binary.preds`
: logical specifying predicted outcome variable will take binary values or proportions.

`bootstrap`
: logical for bootstrapped PATT-C.

`nboot`
: number of bootstrapped samples

Value

`pattc_neural` class object of results of t test as PATTC estimate.

Examples

```
# load datasets
data(exp_data) #experimental data
data(pop_data) #population data
# specify models and estimate PATTC
set.seed(123456)

pattc_neural_boot <- pattc_neural(response.formula = support_war ~ age + female +
                                  income + education +  employed + married +
                                  hindu + job_loss,
                                  exp.data = exp_data,
                                  pop.data = pop_data,
                                  treat.var = "strong_leader",
                                  compl.var = "compliance",
                                  compl.algorithm = "rprop+",
                                  response.algorithm = "rprop+",
                                  compl.hidden.layer = c(2),
                                  response.hidden.layer = c(2),
                                  compl.stepmax = 1e+09,
```

```
                                response.stepmax = 1e+09,
                                ID = NULL,
                                cluster = NULL,
                                binary.preds = FALSE,
                                bootstrap = TRUE,
                                nboot = 1000)
```

`plot.metalearner_ensemble` *plot.metalearner_ensemble*

Description

Uses `plot()` to generate histogram of distribution of CATEs or predicted outcomes from `metalearner_ensemble`

Usage

```
## S3 method for class 'metalearner_ensemble'
plot(x, ..., conf_level = 0.95, type = "CATEs")
```

Arguments

`x`	`metalearner_ensemble` model object
`...`	Additional arguments
`conf_level`	numeric value for confidence level. Defaults to 0.95.
`type`	"CATEs" or "predict"

Value

`ggplot` object

`plot.metalearner_neural` *plot.metalearner_neural*

Description

Uses `plot()` to generate histogram of distribution of CATEs or predicted outcomes from `metalearner_neural`

Usage

```
## S3 method for class 'metalearner_neural'
plot(x, ..., conf_level = 0.95, type = "CATEs")
```

Arguments

x	metalearner_neural model object.
...	Additional arguments
conf_level	numeric value for confidence level. Defaults to 0.95.
type	"CATEs" or "predict".

Value

ggplot object.

---

plot.pattc_deeplearning *plot.pattc_deeplearning*

Description

Uses plot() to generate histogram of distribution of predicted outcomes from pattc_deeplearning

Usage

```
## S3 method for class 'pattc_deeplearning'
plot(x, ...)
```

Arguments

x	pattc_deeplearning model object
...	Additional arguments

Value

ggplot object

---

plot.pattc_ensemble *plot.pattc_ensemble*

Description

Uses plot() to generate histogram of distribution of CATEs or predicted outcomes from pattc_ensemble

Usage

```
## S3 method for class 'pattc_ensemble'
plot(x, ...)
```

Arguments

x	pattc_ensemble model object
...	Additional arguments

Value

ggplot object

---

plot.pattc_neural *plot.pattc_neural*

Description

Uses plot() to generate histogram of distribution of CATEs or predicted outcomes from pattc_neural

Usage

```
## S3 method for class 'pattc_neural'
plot(x, ...)
```

Arguments

x	pattc_neural model object
...	Additional arguments

Value

ggplot object

---

pop_data *World Value Survey India Sample*

Description

World Value Survey (WVS) Data for India in 2022. The variables drawn from the said WVS India data match the covariates from the India survey experiment sample.

Usage

```
data(pop_data)
```

Format

`pop_data`**:**

A data frame with 846 rows and 13 columns:

female Respondent's Sex.

age Age of respondent.

income income group of Household.

religion Religious denomination

practicing_religion Importance of religion in respondent's life.

education Educational level of respondent.

political_ideology Political ideology of respondent.

employment Employment status and full-time employee.

marital_status Marital status of respondent.

job_loss Concern about job loss.

support_war Binary (Yes/No) outcome measure for willingness to fight war.

strong_leader Binary measure of preference for strong leader. ...

Source

Haerpfer, C., Inglehart, R., Moreno, A., Welzel, C., Kizilova, K., Diez-Medrano J., M. Lagos, P. Norris, E. Ponarin & B. Puranen et al. (eds.). 2020. World Values Survey: Round Seven – Country-Pooled Datafile. Madrid, Spain & Vienna, Austria: JD Systems Institute & WVSA Secretariat. <doi.org/10.14281/18241.1>

`pop_data_full` *World Value Survey India Sample*

Description

Extended World Value Survey (WVS) Data for India in 1995, 2001, 2006, 2012, and 2022.

Usage

```
data(pop_data_full)
```

Format

`pop_data_full`**:**

A data frame with 11,813 rows and 13 columns:

female Respondent's Sex.

age Age of respondent.

income income group of Household.

religion Religious denomination

practicing_religion Importance of religion in respondent's life.

education Educational level of respondent.

political_ideology Political ideology of respondent.

employment Employment status and full-time employee.

marital_status Marital status of respondent.

job_loss Concern about job loss.

support_war Binary (Yes/No) outcome measure for willingness to fight war.

strong_leader Binary measure of preference for strong leader. ...

Source

Haerpfer, C., Inglehart, R., Moreno, A., Welzel, C., Kizilova, K., Diez-Medrano J., M. Lagos, P. Norris, E. Ponarin & B. Puranen et al. (eds.). 2020. World Values Survey: Round Seven – Country-Pooled Datafile. Madrid, Spain & Vienna, Austria: JD Systems Institute & WVSA Secretariat. <doi.org/10.14281/18241.1>

`print.metalearner_deeplearning`

print.metalearner_deeplearning

Description

Print method for `metalearner_deeplearning`

Usage

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

Arguments

`x`	`metalearner_deeplearning` class object from `metalearner_deeplearning`
`...`	additional parameter

Value

list of model results

print.metalearner_ensemble *print.metalearner_ensemble*

Description

Print method for `metalearner_ensemble`

Usage

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

Arguments

`x`	`metalearner_ensemble` class object from `metalearner_ensemble`
`...`	additional parameter

Value

list of model results

print.metalearner_neural *print.metalearner_neural*

Description

Print method for `metalearner_neural`

Usage

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

Arguments

`x`	`metalearner_neural` class object from `metalearner_neural`
`...`	additional parameter

Value

list of model results

print.pattc_deeplearning *print.pattc_deeplearning*

Description

Print method for pattc_deeplearning

Usage

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

Arguments

x	pattc_deeplearning class object from pattc_deeplearning
...	additional arguments

Value

list of model results

print.pattc_ensemble *print.pattc_ensemble*

Description

Print method for pattc_ensemble

Usage

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

Arguments

x	pattc_ensemble class object from pattc_ensemble
...	additional parameter

Value

list of model results

print.pattc_neural *print.pattc_neural*

Description

Print method for pattc_neural

Usage

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

Arguments

x	pattc_neural class object from pattc_neural
...	additional parameter

Value

list of model results

python_ready *Check for Python module availability and install if missing.*

Description

Call this to manually set up Python and dependencies. The function checks if Python is available via the reticulate package, and if not, it creates a virtual environment and installs the specified Python modules.

Usage

```
python_ready(
  modules = c("keras", "tensorflow", "numpy"),
  envname = "r-reticulate"
)
```

Arguments

modules	Character vector of Python modules to check for and install if missing.
envname	Name of the virtual environment to use or create. Defaults to "r-reticulate".

Value

Invisibly returns TRUE if setup is complete.

Examples

```
## Not run:
python_ready(modules = c("keras", "tensorflow", "numpy"),
             envname = "r-reticulate")

## End(Not run)
```

`response_model` *Response model from experimental data using SL ensemble*

Description

Train response model (response variable as outcome and covariates) from all compliers (actual + predicted) in experimental data using SL ensemble.

Usage

```
response_model(
  response.formula,
  exp.data,
  compl.var,
  exp.compliers,
  family = gaussian(),
  ID = NULL,
  SL.learners = c("SL.glmnet", "SL.xgboost", "SL.ranger", "SL.nnet", "SL.glm")
)
```

Arguments

`response.formula`	formula to fit the response model (y ~ x) using binary outcome variable and covariates
`exp.data`	experimental dataset.
`compl.var`	string specifying binary complier variable
`exp.compliers`	`data.frame` object of compliers from `complier_predict`.
`family`	gaussian() or binomial().
`ID`	string for identifier variable.
`SL.learners`	vector of names of ML algorithms used for ensemble model.

Value

trained response model.

Index