Package 'easyml'

June 26, 2017

Version 0.1.0

Title Easily Build and Evaluate Machine Learning Models

Description Easily build and evaluate machine learning models on a dataset. Machine learning models supported include penalized linear models, penalized linear models with interactions, random forest, support vector machines, neural networks, and deep neural networks.

License MIT + file LICENSE

LazyData true

Depends R (>= 3.3.1)

Imports caret, corrplot, darch, dummies, e1071, futile.logger, ggplot2, glinternet, glmnet, parallel, pbapply, pbmcapply, pROC, nnet, randomForest, scales, scorer

Suggests covr, lintr, testthat, knitr, rmarkdown

URL https://github.com/CCS-Lab/easyml

BugReports https://github.com/CCS-Lab/easyml/issues

RoxygenNote 6.0.1

NeedsCompilation no

Author Woo-Young Ahn [aut, cre], Paul Hendricks [aut], OSU-CCSL [cph]

Maintainer Woo-Young Ahn <ahn.280@osu.edu>

Repository CRAN

Date/Publication 2017-06-26 06:13:47 UTC

R topics documented:

`cocaine_dependence` *Cocaine data.*

Description

Cocaine data.

Usage

```
cocaine_dependence
```

Format

Data frame with columns

subject Subject ID

diagnosis Diagnosis status

age Age of Subject

male Gender of subject

edu_yrs Education of subject in years

imt_comm_errors Immediate Memory Task commission errors

imt_omis_errors Immediate Memory Task omission errors

a_imt Immediate Memory Task non-parametric discriminability

b_d_imt Immediate Memory Task response bias

stop_ssrt Stop-Signal Task stop-signal reaction time

lnk_adjdd log(Adjusting Delay-Discounting Task discounting rate)

lkhat_kirby log(Kirby Monetary-Choice Delay-Discounting Questionnaire)

revlr_per_errors Probabilistic Reversal-Learning Task errors

bis_attention Barratt Impulsiveness Scale attentional impulsivity

bis_motor Barratt Impulsiveness Scale motor impulsivity

bis_nonpL Barratt Impulsiveness Scale nonplanning impulsivity

igt_total Iowa Gambling Task total score

Source

https://github.com/CCS-Lab/easyml/blob/master/Python/datasets/cocaine_dependence.csv

See Also

Other data: `prostate`

Examples

```
data("cocaine_dependence", package = "easyml")
```

`correlation_test` *Compute the matrix of p-value.*

Description

See here for source: http://www.sthda.com/english/wiki/visualize-correlation-matrix-using-correlogram.

Usage

```
correlation_test(x, confidence_level = 0.95, ...)
```

Arguments

`x`	An object of class "DataFrame" or "matrix".
`confidence_level`	confidence level for the returned confidence interval. Currently only used for the Pearson product moment correlation coefficient if there are at least 4 complete pairs of observations.
`...`	further arguments to be passed to or from `cor.test`.

Value

A list containing three matrices; p_value, lower_bound, and upper bound.

See Also

Other utils: process_coefficients, process_variable_importances, reduce_cores, remove_variables

---

easyml *easyml: Easily build and evaluate machine learning models.*

Description

easyml: Easily build and evaluate machine learning models.

---

easy_analysis *The core recipe of easyml.*

Description

This recipe is the workhorse behind all of the easy_* functions.

Usage

```
easy_analysis(.data, dependent_variable, algorithm, family = "gaussian",
  resample = NULL, preprocess = NULL, measure = NULL,
  exclude_variables = NULL, categorical_variables = NULL,
  train_size = 0.667, foldid = NULL, survival_rate_cutoff = 0.05,
  n_samples = 1000, n_divisions = 1000, n_iterations = 10,
  random_state = NULL, progress_bar = TRUE, n_core = 1,
  coefficients = NULL, variable_importances = NULL, predictions = NULL,
  model_performance = NULL, model_args = list())
```

Arguments

.data	A data.frame; the data to be analyzed.
dependent_variable	A character vector of length one; the dependent variable for this analysis.
algorithm	A character vector of length one; the algorithm to run on the data. Choices are currently one of c("deep_neural_network", "glinternet", "glmnet", "neural_network", "random_forest", "support_vector_machine").
family	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
resample	A function; the function for resampling the data. Defaults to NULL.
preprocess	A function; the function for preprocessing the data. Defaults to NULL.

`measure` A function; the function for measuring the results. Defaults to NULL.

`exclude_variables`
A character vector; the variables from the data set to exclude. Defaults to NULL.

`categorical_variables`
A character vector; the variables that are categorical. Defaults to NULL.

`train_size` A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.

`foldid` A vector with length equal to `length(y)` which identifies cases belonging to the same fold.

`survival_rate_cutoff`
A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.

`n_samples` An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.

`n_divisions` An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.

`n_iterations` An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.

`random_state` An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

`progress_bar` A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.

`n_core` An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.

`coefficients` A logical vector of length one; whether or not to generate coefficients for this analysis.

`variable_importances`
A logical vector of length one; whether or not to generate variable importances for this analysis.

`predictions` A logical vector of length one; whether or not to generate predictions for this analysis.

`model_performance`
A logical vector of length one; whether or not to generate measures of model performance for this analysis.

`model_args` A list; the arguments to be passed to the algorithm specified.

Value

A list of class `easy_*`, where * is the name of the algorithm.

call An object of class `call`; the original function call.

data A data.frame; the original data.

dependent_variable A character vector of length one; the dependent variable for this analysis.

algorithm A character vector of length one; the algorithm to run on the data.

class A character vector of length one; the class of the object.

family A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".

resample A function; the function for resampling the data.

preprocess A function; the function for preprocessing the data.

measure A function; the function for measuring the results.

exclude_variables A character vector; the variables from the data set to exclude.

train_size A numeric vector of length one; specifies what proportion of the data should be used for the training data set.

survival_rate_cutoff A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples.

n_samples An integer vector of length one; specifies the number of times the coefficients and predictions should be generated.

n_divisions An integer vector of length one; specifies the number of times the data should be divided when generating measures of model performance.

n_iterations An integer vector of length one; during each division, specifies the number of times the predictions should be generated.

random_state An integer vector of length one; specifies the seed to be used for the analysis.

progress_bar A logical vector of length one; specifies whether to display a progress bar during calculations.

n_core An integer vector of length one; specifies the number of cores to use for this analysis.

generate_coefficients A logical vector of length one; whether or not to generate coefficients for this analysis.

generate_variable_importances A logical vector of length one; whether or not to generate variable importances for this analysis.

generate_predictions A logical vector of length one; whether or not to generate predictions for this analysis.

generate_model_performance A logical vector of length one; whether or not to generate measures of model performance for this analysis.

model_args A list; the arguments to be passed to the algorithm specified.

column_names A character vector; the column names.

categorical_variables A logical vector; the variables that are categorical.

X A data.frame; the full dataset to be used for modeling.

y A vector; the full response variable to be used for modeling.

coefficients A (n_variables, n_samples) matrix; the generated coefficients.

coefficients_processed A data.frame; the coefficients after being processed.

plot_coefficients_processed A ggplot object; the plot of the processed coefficients.

X_train A data.frame; the train dataset to be used for modeling.

X_test A data.frame; the test dataset to be used for modeling.

y_train A vector; the train response variable to be used for modeling.

y_test A vector; the test response variable to be used for modeling.

predictions_train A (nrow(X_train), n_samples) matrix; the train predictions.

predictions_test A (nrow(X_test), n_samples) matrix; the test predictions.

predictions_train_mean A vector; the mean train predictions.

predictions_test_mean A vector; the mean test predictions.

plot_predictions A function; the function for plotting predictions generated by the model.

plot_predictions_train_mean A ggplot object; the plot of the mean train predictions.

plot_predictions_test_mean A ggplot object; the plot of the mean test predictions.

model_performance_train A vector of length n_divisions; the measures of model performance on the train datasets.

model_performance_test A vector of length n_divisions; the measures of model performance on the test datasets.

plot_model_performance A function; the function for plotting the measures of model performance.

plot_model_performance_train A ggplot object; the plot of the measures of model performance on the train datasets.

plot_model_performance_test A ggplot object; the plot of the measures of model performance on the test datasets.

See Also

Other recipes: easy_avNNet, easy_deep_neural_network, easy_glinternet, easy_glmnet, easy_neural_network, easy_random_forest, easy_support_vector_machine

---

easy_avNNet *Easily build and evaluate an average neural network model.*

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for an average neural network model.

Usage

```
easy_avNNet(.data, dependent_variable, family = "gaussian", resample = NULL,
  preprocess = preprocess_scale, measure = NULL, exclude_variables = NULL,
  categorical_variables = NULL, train_size = 0.667, foldid = NULL,
  survival_rate_cutoff = 0.05, n_samples = 1000, n_divisions = 1000,
  n_iterations = 10, random_state = NULL, progress_bar = TRUE,
  n_core = 1, coefficients = FALSE, variable_importances = FALSE,
  predictions = TRUE, model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.
`preprocess`	A function; the function for preprocessing the data. Defaults to NULL.
`measure`	A function; the function for measuring the results. Defaults to NULL.
`exclude_variables`	A character vector; the variables from the data set to exclude. Defaults to NULL.
`categorical_variables`	A character vector; the variables that are categorical. Defaults to NULL.
`train_size`	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
`foldid`	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.
`survival_rate_cutoff`	A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.
`n_samples`	An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.
`n_divisions`	An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.
`n_iterations`	An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.
`random_state`	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.
`progress_bar`	A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.
`n_core`	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.
`coefficients`	A logical vector of length one; whether or not to generate coefficients for this analysis.
`variable_importances`	A logical vector of length one; whether or not to generate variable importances for this analysis.
`predictions`	A logical vector of length one; whether or not to generate predictions for this analysis.
`model_performance`	A logical vector of length one; whether or not to generate measures of model performance for this analysis.
`model_args`	A list; the arguments to be passed to the algorithm specified.

Value

A list of class `easy_avNNet`.

See Also

Other recipes: `easy_analysis`, `easy_deep_neural_network`, `easy_glinternet`, `easy_glmnet`, `easy_neural_network`, `easy_random_forest`, `easy_support_vector_machine`

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
data("prostate", package = "easyml")
results <- easy_avNNet(prostate, "lpsa",
                       n_samples = 10, n_divisions = 10,
                       n_iterations = 2, random_state = 12345,
                       n_core = 1)

# Binomial
data("cocaine_dependence", package = "easyml")
model_args <- list(size = 5, linout = TRUE, trace = FALSE)
results <- easy_avNNet(cocaine_dependence, "diagnosis",
                       family = "binomial",
                       exclude_variables = c("subject"),
                       categorical_variables = c("male"),
                       preprocess = preprocess_scale,
                       n_samples = 10, n_divisions = 10,
                       n_iterations = 2, random_state = 12345,
                       n_core = 1, model_args = model_args)

## End(Not run)
```

---

easy_deep_neural_network

Easily build and evaluate a deep neural network.

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for a deep neural network model.

Usage

```
easy_deep_neural_network(.data, dependent_variable, family = "gaussian",
  resample = NULL, preprocess = preprocess_scale, measure = NULL,
  exclude_variables = NULL, categorical_variables = NULL,
  train_size = 0.667, foldid = NULL, survival_rate_cutoff = 0.05,
```

```
  n_samples = 1000, n_divisions = 1000, n_iterations = 10,
  random_state = NULL, progress_bar = TRUE, n_core = 1,
  coefficients = FALSE, variable_importances = FALSE, predictions = TRUE,
  model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.
`preprocess`	A function; the function for preprocessing the data. Defaults to NULL.
`measure`	A function; the function for measuring the results. Defaults to NULL.
`exclude_variables`	A character vector; the variables from the data set to exclude. Defaults to NULL.
`categorical_variables`	A character vector; the variables that are categorical. Defaults to NULL.
`train_size`	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
`foldid`	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.
`survival_rate_cutoff`	A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.
`n_samples`	An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.
`n_divisions`	An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.
`n_iterations`	An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.
`random_state`	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.
`progress_bar`	A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.
`n_core`	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.
`coefficients`	A logical vector of length one; whether or not to generate coefficients for this analysis.
`variable_importances`	A logical vector of length one; whether or not to generate variable importances for this analysis.

predictions A logical vector of length one; whether or not to generate predictions for this analysis.

model_performance A logical vector of length one; whether or not to generate measures of model performance for this analysis.

model_args A list; the arguments to be passed to the algorithm specified.

Value

A list of class easy_deep_neural_network.

See Also

Other recipes: easy_analysis, easy_avNNet, easy_glinternet, easy_glmnet, easy_neural_network, easy_random_forest, easy_support_vector_machine

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
data("prostate", package = "easyml")
results <- easy_deep_neural_network(prostate, "lpsa",
                                    n_samples = 10, n_divisions = 10,
                                    n_iterations = 2, random_state = 12345,
                                    n_core = 1)

# Binomial
data("cocaine_dependence", package = "easyml")
results <- easy_deep_neural_network(cocaine_dependence, "diagnosis",
                                    family = "binomial",
                                    exclude_variables = c("subject"),
                                    categorical_variables = c("male"),
                                    preprocess = preprocess_scale,
                                    n_samples = 10, n_divisions = 10,
                                    n_iterations = 2, random_state = 12345,
                                    n_core = 1)

## End(Not run)
```

---

easy_glinternet *Easily build and evaluate a penalized regression model with interactions.*

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for a glinternet model.

Usage

```
easy_glinternet(.data, dependent_variable, family = "gaussian",
  resample = NULL, preprocess = preprocess_scale, measure = NULL,
  exclude_variables = NULL, categorical_variables = NULL,
  train_size = 0.667, foldid = NULL, survival_rate_cutoff = 0.05,
  n_samples = 1000, n_divisions = 1000, n_iterations = 10,
  random_state = NULL, progress_bar = TRUE, n_core = 1,
  coefficients = FALSE, variable_importances = FALSE, predictions = TRUE,
  model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.
`preprocess`	A function; the function for preprocessing the data. Defaults to NULL.
`measure`	A function; the function for measuring the results. Defaults to NULL.
`exclude_variables`	A character vector; the variables from the data set to exclude. Defaults to NULL.
`categorical_variables`	A character vector; the variables that are categorical. Defaults to NULL.
`train_size`	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
`foldid`	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.
`survival_rate_cutoff`	A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.
`n_samples`	An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.
`n_divisions`	An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.
`n_iterations`	An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.
`random_state`	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.
`progress_bar`	A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.
`n_core`	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.

`coefficients` A logical vector of length one; whether or not to generate coefficients for this analysis.

`variable_importances` A logical vector of length one; whether or not to generate variable importances for this analysis.

`predictions` A logical vector of length one; whether or not to generate predictions for this analysis.

`model_performance` A logical vector of length one; whether or not to generate measures of model performance for this analysis.

`model_args` A list; the arguments to be passed to the algorithm specified.

Value

A list of class `easy_glinternet`.

See Also

Other recipes: `easy_analysis`, `easy_avNNet`, `easy_deep_neural_network`, `easy_glmnet`, `easy_neural_network`, `easy_random_forest`, `easy_support_vector_machine`

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
data("prostate", package = "easyml")
results <- easy_glinternet(prostate, "lpsa",
                           n_samples = 10, n_divisions = 10,
                           n_iterations = 2, random_state = 12345,
                           n_core = 1)

# Binomial
data("cocaine_dependence", package = "easyml")
results <- easy_glinternet(cocaine_dependence, "diagnosis",
                           family = "binomial",
                           exclude_variables = c("subject"),
                           categorical_variables = c("male"),
                           preprocess = preprocess_scale,
                           n_samples = 10, n_divisions = 10,
                           n_iterations = 2, random_state = 12345,
                           n_core = 1)

## End(Not run)
```

`easy_glmnet` *Easily build and evaluate a penalized regression model.*

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for a glmnet model.

Usage

```
easy_glmnet(.data, dependent_variable, family = "gaussian", resample = NULL,
  preprocess = preprocess_scale, measure = NULL, exclude_variables = NULL,
  categorical_variables = NULL, train_size = 0.667, foldid = NULL,
  survival_rate_cutoff = 0.05, n_samples = 1000, n_divisions = 1000,
  n_iterations = 10, random_state = NULL, progress_bar = TRUE,
  n_core = 1, coefficients = TRUE, variable_importances = FALSE,
  predictions = TRUE, model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.
`preprocess`	A function; the function for preprocessing the data. Defaults to NULL.
`measure`	A function; the function for measuring the results. Defaults to NULL.
`exclude_variables`	A character vector; the variables from the data set to exclude. Defaults to NULL.
`categorical_variables`	A character vector; the variables that are categorical. Defaults to NULL.
`train_size`	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
`foldid`	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.
`survival_rate_cutoff`	A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.
`n_samples`	An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.
`n_divisions`	An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.

`n_iterations` An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.

`random_state` An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

`progress_bar` A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.

`n_core` An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.

`coefficients` A logical vector of length one; whether or not to generate coefficients for this analysis.

`variable_importances`
A logical vector of length one; whether or not to generate variable importances for this analysis.

`predictions` A logical vector of length one; whether or not to generate predictions for this analysis.

`model_performance`
A logical vector of length one; whether or not to generate measures of model performance for this analysis.

`model_args` A list; the arguments to be passed to the algorithm specified.

Value

A list of class `easy_glmnet`.

See Also

Other recipes: `easy_analysis`, `easy_avNNet`, `easy_deep_neural_network`, `easy_glinternet`, `easy_neural_network`, `easy_random_forest`, `easy_support_vector_machine`

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
data("prostate", package = "easyml")
results <- easy_glmnet(prostate, "lpsa",
                       n_samples = 10, n_divisions = 10,
                       n_iterations = 2, random_state = 12345,
                       n_core = 1, model_args = list(alpha = 1.0))

# Binomial
data("cocaine_dependence", package = "easyml")
results <- easy_glmnet(cocaine_dependence, "diagnosis",
                       family = "binomial",
                       exclude_variables = c("subject"),
                       categorical_variables = c("male"),
```

```
                        preprocess = preprocess_scale,
                        n_samples = 10, n_divisions = 10,
                        n_iterations = 2, random_state = 12345,
                        n_core = 1, model_args = list(alpha = 1.0))

## End(Not run)
```

`easy_neural_network` *Easily build and evaluate a neural network.*

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for a neural network model.

Usage

```
easy_neural_network(.data, dependent_variable, family = "gaussian",
  resample = NULL, preprocess = preprocess_scale, measure = NULL,
  exclude_variables = NULL, categorical_variables = NULL,
  train_size = 0.667, foldid = NULL, survival_rate_cutoff = 0.05,
  n_samples = 1000, n_divisions = 1000, n_iterations = 10,
  random_state = NULL, progress_bar = TRUE, n_core = 1,
  coefficients = FALSE, variable_importances = FALSE, predictions = TRUE,
  model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.
`preprocess`	A function; the function for preprocessing the data. Defaults to NULL.
`measure`	A function; the function for measuring the results. Defaults to NULL.
`exclude_variables`	A character vector; the variables from the data set to exclude. Defaults to NULL.
`categorical_variables`	A character vector; the variables that are categorical. Defaults to NULL.
`train_size`	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
`foldid`	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.

survival_rate_cutoff
: A numeric vector of length one; for easy_glmnet, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.

n_samples
: An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.

n_divisions
: An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.

n_iterations
: An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.

random_state
: An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

progress_bar
: A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.

n_core
: An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.

coefficients
: A logical vector of length one; whether or not to generate coefficients for this analysis.

variable_importances
: A logical vector of length one; whether or not to generate variable importances for this analysis.

predictions
: A logical vector of length one; whether or not to generate predictions for this analysis.

model_performance
: A logical vector of length one; whether or not to generate measures of model performance for this analysis.

model_args
: A list; the arguments to be passed to the algorithm specified.

Value

A list of class easy_neural_network.

See Also

Other recipes: easy_analysis, easy_avNNet, easy_deep_neural_network, easy_glinternet, easy_glmnet, easy_random_forest, easy_support_vector_machine

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
data("prostate", package = "easyml")
results <- easy_neural_network(prostate, "lpsa",
                               n_samples = 10, n_divisions = 10,
```

```
                                n_iterations = 2, random_state = 12345,
                                n_core = 1)

# Binomial
data("cocaine_dependence", package = "easyml")
results <- easy_neural_network(cocaine_dependence, "diagnosis",
                                family = "binomial",
                                exclude_variables = c("subject"),
                                categorical_variables = c("male"),
                                preprocess = preprocess_scale,
                                n_samples = 10, n_divisions = 10,
                                n_iterations = 2, random_state = 12345,
                                n_core = 1)

## End(Not run)
```

`easy_random_forest` *Easily build and evaluate a random forest regression model.*

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for a random forest model.

Usage

```
easy_random_forest(.data, dependent_variable, family = "gaussian",
  resample = NULL, preprocess = preprocess_identity, measure = NULL,
  exclude_variables = NULL, categorical_variables = NULL,
  train_size = 0.667, foldid = NULL, n_samples = 1000,
  n_divisions = 1000, n_iterations = 10, random_state = NULL,
  progress_bar = TRUE, n_core = 1, coefficients = FALSE,
  variable_importances = TRUE, predictions = TRUE,
  model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.
`preprocess`	A function; the function for preprocessing the data. Defaults to NULL.
`measure`	A function; the function for measuring the results. Defaults to NULL.
`exclude_variables`	A character vector; the variables from the data set to exclude. Defaults to NULL.

categorical_variables
: A character vector; the variables that are categorical. Defaults to NULL.

train_size
: A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.

foldid
: A vector with length equal to `length(y)` which identifies cases belonging to the same fold.

n_samples
: An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.

n_divisions
: An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.

n_iterations
: An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.

random_state
: An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

progress_bar
: A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.

n_core
: An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.

coefficients
: A logical vector of length one; whether or not to generate coefficients for this analysis.

variable_importances
: A logical vector of length one; whether or not to generate variable importances for this analysis.

predictions
: A logical vector of length one; whether or not to generate predictions for this analysis.

model_performance
: A logical vector of length one; whether or not to generate measures of model performance for this analysis.

model_args
: A list; the arguments to be passed to the algorithm specified.

Value

A list of class `easy_random_forest`.

See Also

Other recipes: `easy_analysis`, `easy_avNNet`, `easy_deep_neural_network`, `easy_glinternet`, `easy_glmnet`, `easy_neural_network`, `easy_support_vector_machine`

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
```

```
data("prostate", package = "easyml")
results <- easy_random_forest(prostate, "lpsa",
                              n_samples = 10L,
                              n_divisions = 10,
                              n_iterations = 2,
                              random_state = 12345, n_core = 1)

# Binomial
data("cocaine_dependence", package = "easyml")
results <- easy_random_forest(cocaine_dependence, "diagnosis",
                              family = "binomial",
                              exclude_variables = c("subject"),
                              categorical_variables = c("male"),
                              n_samples = 10,
                              n_divisions = 10,
                              n_iterations = 2,
                              random_state = 12345, n_core = 1)

## End(Not run)
```

`easy_support_vector_machine` *Easily build and evaluate a support vector machine regression model.*

Description

This function wraps the easyml core framework, allowing a user to easily run the easyml methodology for a support vector machine model.

Usage

```
easy_support_vector_machine(.data, dependent_variable, family = "gaussian",
  resample = NULL, preprocess = preprocess_scale, measure = NULL,
  exclude_variables = NULL, categorical_variables = NULL,
  train_size = 0.667, foldid = NULL, n_samples = 1000,
  n_divisions = 1000, n_iterations = 10, random_state = NULL,
  progress_bar = TRUE, n_core = 1, coefficients = FALSE,
  variable_importances = FALSE, predictions = TRUE,
  model_performance = TRUE, model_args = list())
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".
`resample`	A function; the function for resampling the data. Defaults to NULL.

`preprocess` A function; the function for preprocessing the data. Defaults to NULL.

`measure` A function; the function for measuring the results. Defaults to NULL.

`exclude_variables`
A character vector; the variables from the data set to exclude. Defaults to NULL.

`categorical_variables`
A character vector; the variables that are categorical. Defaults to NULL.

`train_size` A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.

`foldid` A vector with length equal to `length(y)` which identifies cases belonging to the same fold.

`n_samples` An integer vector of length one; specifies the number of times the coefficients and predictions should be generated. Defaults to 1000.

`n_divisions` An integer vector of length one; specifies the number of times the data should be divided when replicating the measures of model performance. Defaults to 1000.

`n_iterations` An integer vector of length one; during each division, specifies the number of times the predictions should be generated. Defaults to 10.

`random_state` An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

`progress_bar` A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.

`n_core` An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1.

`coefficients` A logical vector of length one; whether or not to generate coefficients for this analysis.

`variable_importances`
A logical vector of length one; whether or not to generate variable importances for this analysis.

`predictions` A logical vector of length one; whether or not to generate predictions for this analysis.

`model_performance`
A logical vector of length one; whether or not to generate measures of model performance for this analysis.

`model_args` A list; the arguments to be passed to the algorithm specified.

Value

A list of class `easy_support_vector_machine`.

See Also

Other recipes: `easy_analysis`, `easy_avNNet`, `easy_deep_neural_network`, `easy_glinternet`, `easy_glmnet`, `easy_neural_network`, `easy_random_forest`

Examples

```
## Not run:
library(easyml) # https://github.com/CCS-Lab/easyml

# Gaussian
data("prostate", package = "easyml")
results <- easy_support_vector_machine(prostate, "lpsa",
                                       n_samples = 10,
                                       n_divisions = 10,
                                       n_iterations = 2,
                                       random_state = 1, n_core = 1)

# Binomial
data("cocaine_dependence", package = "easyml")
results <- easy_support_vector_machine(cocaine_dependence, "diagnosis",
                                       family = "binomial",
                                       preprocesss = preprocess_scale,
                                       exclude_variables = c("subject"),
                                       categorical_variables = c("male"),
                                       n_samples = 10,
                                       n_divisions = 10,
                                       n_iterations = 2,
                                       random_state = 1, n_core = 1)

## End(Not run)
```

---

extract_coefficients *Extract coefficients.*

Description

The generic function for extracting coefficients from a model within the easyml core framework, if such an operation is applicable for that model. Users can create their own extract_coefficients function for their own class of model.

Usage

```
extract_coefficients(object)
```

Arguments

object	A list of class easy_*, where * is the name of the algorithm.

Value

A data.frame, the generated coefficients.

`extract_coefficients.easy_glmnet` *Extract coefficients from a penalized regression model.*

Description

This function wraps the procedure for extracting coefficients from a glmnet model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_glmnet'
extract_coefficients(object)
```

Arguments

`object`	A list of class `easy_glmnet`.

Value

A data.frame, the replicated penalized regression coefficients.

`extract_variable_importances` *Extract variable importances.*

Description

The generic function for extracting variable importances from a model within the easyml core framework, if such an operation is applicable for that model. Users can create their own extract_variable_importances function for their own class of model.

Usage

```
extract_variable_importances(object)
```

Arguments

`object`	A list of class `easy_*`, where * is the name of the algorithm.

Value

A data.frame, the generated random forest variable importance scores.

extract_variable_importances.easy_random_forest

Extract variable importance scores from a random forest model.

Description

This function wraps the procedure for extracting variable importances from a random forest model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_random_forest'
extract_variable_importances(object)
```

Arguments

`object`	A list of class `easy_random_forest`.

Value

A data.frame, the replicated random forest variable importance scores.

fit_model

Fit model.

Description

The generic function for fitting a model within the easyml core framework. Users can create their own fit_model function for their own class of model.

Usage

```
fit_model(object)
```

Arguments

`object`	A list of class `easy_*`, where * is the name of the algorithm.

Value

A list of class `easy_*`, where * is the name of the algorithm.

fit_model.easy_avNNet *Fit an average neural network model.*

Description

This function wraps the procedure for fitting an average neural network model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_avNNet'
fit_model(object)
```

Arguments

object	A list of class easy_avNNet.

Value

A list of class easy_avNNet.

fit_model.easy_deep_neural_network *Fit a deep neural network model.*

Description

This function wraps the procedure for fitting a deep neural network model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_deep_neural_network'
fit_model(object)
```

Arguments

object	A list of class easy_deep_neural_network.

Value

A list of class easy_deep_neural_network.

fit_model.easy_glinternet *Fit a penalized regression model with interactions.*

Description

This function wraps the procedure for fitting a glinternet model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_glinternet'
fit_model(object)
```

Arguments

object	A list of class easy_glinternet.

Value

A list of class easy_glinternet.

fit_model.easy_glmnet *Fit a penalized regression model.*

Description

This function wraps the procedure for fitting a glmnet model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_glmnet'
fit_model(object)
```

Arguments

object	A list of class easy_glmnet.

Value

A list of class easy_glmnet.

`fit_model.easy_neural_network`
Fit a neural network model.

Description

This function wraps the procedure for fitting a neural network model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_neural_network'
fit_model(object)
```

Arguments

`object`	A list of class `easy_neural_network`.

Value

A list of class `easy_neural_network`.

`fit_model.easy_random_forest`
Fit a random forest model.

Description

This function wraps the procedure for fitting a random forest model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_random_forest'
fit_model(object)
```

Arguments

`object`	A list of class `easy_random_forest`.

Value

A list of class `easy_random_forest`.

fit_model.easy_support_vector_machine *Fit a support vector machine regression model.*

Description

This function wraps the procedure for fitting a support vector machine model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_support_vector_machine'
fit_model(object)
```

Arguments

object	A list of class easy_support_vector_machine.

Value

A list of class easy_support_vector_machine.

generate_coefficients *Generate coefficients for a model (if applicable).*

Description

Generate coefficients for a model (if applicable).

Usage

```
generate_coefficients(object)
```

Arguments

object	A list of class easy_*, where * is the name of the algorithm.

Value

A data.frame, the generated penalized regression model coefficients.

See Also

Other generate: generate_model_performance, generate_predictions, generate_variable_importances

generate_model_performance *Generate measures of model performance for a model.*

Description

Generate measures of model performance for a model.

Usage

```
generate_model_performance(object)
```

Arguments

object	A list of class easy_*, where * is the name of the algorithm.

Value

A list of matrixes, the generated measures of model performance.

See Also

Other generate: generate_coefficients, generate_predictions, generate_variable_importances

generate_predictions *Generate predictions for a model.*

Description

Generate predictions for a model.

Usage

```
generate_predictions(object)
```

Arguments

object	A list of class easy_*, where * is the name of the algorithm.

Value

A list of matrixes, the generated predictions.

See Also

Other generate: generate_coefficients, generate_model_performance, generate_variable_importances

generate_variable_importances *Generate variable importances for a model (if applicable).*

Description

Generate variable importances for a model (if applicable).

Usage

```
generate_variable_importances(object)
```

Arguments

object	A list of class easy_*, where * is the name of the algorithm.

Value

A data.frame, the generated variable importance scores.

See Also

Other generate: generate_coefficients, generate_model_performance, generate_predictions

measure_auc_score *Measure area under the curve.*

Description

Given the ground truth (correct) target values and the estimated target values, calculates the the AUC metric.

Usage

```
measure_auc_score(y_true, y_pred)
```

Arguments

y_true	A numeric vector; the ground truth (correct) target values.
y_pred	A numeric vector; the estimated target values.

Value

A numeric vector of length one; the AUC metric.

See Also

Other measure: measure_correlation_score, measure_mse_score, measure_r2_score

measure_correlation_score *Measure Pearsons Correlation Coefficient.*

Description

Given the ground truth (correct) target values and the estimated target values, calculates the correlation metric.

Usage

```
measure_correlation_score(y_true, y_pred)
```

Arguments

y_true	A numeric vector; the ground truth (correct) target values.
y_pred	A numeric vector; the estimated target values.

Details

See here for more information: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

Value

A numeric vector of length one; the correlation metric.

See Also

Other measure: measure_auc_score, measure_mse_score, measure_r2_score

measure_mse_score *Measure mean squared error.*

Description

Given the ground truth (correct) target values and the estimated target values, calculates the mean squared error metric.

Usage

```
measure_mse_score(y_true, y_pred)
```

Arguments

y_true	A numeric vector; the ground truth (correct) target values.
y_pred	A numeric vector; the estimated target values.

Value

A numeric vector of length one; the mean squared error metric.

See Also

Other measure: measure_auc_score, measure_correlation_score, measure_r2_score

---

`measure_r2_score` *Measure Coefficient of Determination (R^2 Score).*

Description

Given the ground truth (correct) target values and the estimated target values, calculates the the R^2 metric.

Usage

```
measure_r2_score(y_true, y_pred)
```

Arguments

y_true	A numeric vector; the ground truth (correct) target values.
y_pred	A numeric vector; the estimated target values.

Details

See here for more information: https://en.wikipedia.org/wiki/Coefficient_of_determination

Value

A numeric vector of length one; the R^2 metric.

See Also

Other measure: measure_auc_score, measure_correlation_score, measure_mse_score

plot_coefficients_processed *Plot penalized regression coefficients.*

Description

When calling easy_glmnet, coefficients from the generate_coefficients output are processed by the process_coefficients function and generated into a plot. This plot tells us the direction, magnitude, and statistical significance of each coefficient. Be careful using this plotting method with datasets containing more than 20 variables as the plot may not render as nicely.

Usage

```
plot_coefficients_processed(coefficients_processed)
```

Arguments

coefficients_processed
: A data.frame, the output of the function process_coefficients.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_correlation_scor
plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score,
plot_model_performance_histogram, plot_predictions_binomial, plot_predictions_gaussian,
plot_roc_curve, plot_variable_importances_processed

plot_model_performance_binomial_auc_score *Plot histogram of the area under the curve (AUC) metrics.*

Description

This function plots a histogram of the area under the curve (AUC) metrics.

Usage

```
plot_model_performance_binomial_auc_score(x)
```

Arguments

x	A vector, the area under the curve (AUC) metrics to be plotted as a histogram.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_gaussian_correlation_score, plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score, plot_model_performance_histogram, plot_predictions_binomial, plot_predictions_gaussian, plot_roc_curve, plot_variable_importances_processed

---

plot_model_performance_gaussian_correlation_score

Plot histogram of the correlation coefficient metrics.

Description

This function plots a histogram of the correlation coefficient metrics.

Usage

```
plot_model_performance_gaussian_correlation_score(x)
```

Arguments

x	A vector, the correlation coefficient metrics to be plotted as a histogram.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score, plot_model_performance_histogram, plot_predictions_binomial, plot_predictions_gaussian, plot_roc_curve, plot_variable_importances_processed

---

`plot_model_performance_gaussian_mse_score`
Plot histogram of the mean squared error metrics.

Description

This function plots a histogram of the mean squared error metrics.

Usage

```
plot_model_performance_gaussian_mse_score(x)
```

Arguments

x	A vector, the mean squared error metrics to be plotted as a histogram.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: `plot_coefficients_processed`, `plot_model_performance_binomial_auc_score`, `plot_model_performance_gaussian_correlation_score`, `plot_model_performance_gaussian_r2_score`, `plot_model_performance_histogram`, `plot_predictions_binomial`, `plot_predictions_gaussian`, `plot_roc_curve`, `plot_variable_importances_processed`

`plot_model_performance_gaussian_r2_score`
Plot histogram of the coefficient of determination (R^2) metrics.

Description

This function plots a histogram of the coefficient of determination (R^2) metrics.

Usage

```
plot_model_performance_gaussian_r2_score(x)
```

Arguments

x	A vector, the coefficient of determination (R^2) metrics to be plotted as a histogram.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_correlation_score, plot_model_performance_gaussian_mse_score, plot_model_performance_histogram, plot_predictions_binomial, plot_predictions_gaussian, plot_roc_curve, plot_variable_importances_processed

---

`plot_model_performance_histogram`

Plot histogram of measures of model performance.

Description

Given a numeric vector will plot a histogram for any number of measures of model performance.

Usage

```
plot_model_performance_histogram(x, name)
```

Arguments

x	A vector, the mean squared error metrics to be plotted as a histogram.
name	A character vector of length one, the name of the metric.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_correlation_score, plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score, plot_predictions_binomial, plot_predictions_gaussian, plot_roc_curve, plot_variable_importances_processed

---

`plot_predictions_binomial`
Plot binomial predictions.

Description

Plots a logistic plot of the ground truth (correct) target values and the estimated target values.

Usage

```
plot_predictions_binomial(y_true, y_pred)
```

Arguments

`y_true`	Ground truth (correct) target values.
`y_pred`	Estimated target values.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: `plot_coefficients_processed`, `plot_model_performance_binomial_auc_score`, `plot_model_performance_gaussian_correlation_score`, `plot_model_performance_gaussian_mse_score`, `plot_model_performance_gaussian_r2_score`, `plot_model_performance_histogram`, `plot_predictions_gaussian`, `plot_roc_curve`, `plot_variable_importances_processed`

`plot_predictions_gaussian`
Plot gaussian predictions.

Description

Plots a scatter plot of the ground truth (correct) target values and the estimated target values.

Usage

```
plot_predictions_gaussian(y_true, y_pred)
```

Arguments

`y_true`	Ground truth (correct) target values.
`y_pred`	Estimated target values.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_correlation_score, plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score, plot_model_performance_histogram, plot_predictions_binomial, plot_roc_curve, plot_variable_importances_processed

---

plot_roc_curve *Plot ROC Curve.*

Description

Given the ground truth (correct) target values and the estimated target values will plot an ROC curve.

Usage

```
plot_roc_curve(y_true, y_pred)
```

Arguments

y_true	Ground truth (correct) target values.
y_pred	Estimated target values.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_correlation_score, plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score, plot_model_performance_histogram, plot_predictions_binomial, plot_predictions_gaussian, plot_variable_importances_processed

plot_variable_importances_processed *Plot random forest variable importances scores.*

Description

When calling easy_random_forest, variable importances scores from the generate_variable_importances output are processed by the process_variable_importances function and generated into a plot. Importance scores for each predictor were estimated using the increase in node impurity. Node impurity measures the change in residual squared error that is attributable to the predictor across all trees. Unlike the easy_glmnet coefficients, random forest importance scores do not indicate directional effects, but instead represent the magnitude of the effect that the predictor has on overall prediction accuracy. Be careful using this plotting method with datasets containing more than 20 variables as the plot may not render as nicely.

Usage

```
plot_variable_importances_processed(variable_importances_processed)
```

Arguments

variable_importances_processed
: A data.frame, the output of the function process_variable_importances.

Value

A ggplot object. This plot may be rendered by outputting it to the command line or modified using ggplot semantics.

See Also

Other plot: plot_coefficients_processed, plot_model_performance_binomial_auc_score, plot_model_performance_gaussian_correlation_score, plot_model_performance_gaussian_mse_score, plot_model_performance_gaussian_r2_score, plot_model_performance_histogram, plot_predictions_binomial, plot_predictions_gaussian, plot_roc_curve

predict_model *Predict model.*

Description

The generic function for generating predictions from a model within the easyml core framework. Users can create their own predict_model function for their own class of model.

Usage

```
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_*, where * is the name of the algorithm.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

---

predict_model.easy_avNNet

Predict values for an average neural network model.

Description

This function wraps the procedure for predicting values from a average neural network model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_avNNet'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_avNNet.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

predict_model.easy_deep_neural_network

Predict values for a deep neural network model.

Description

This function wraps the procedure for predicting values from a deep neural network model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_deep_neural_network'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_deep_neural_network.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

predict_model.easy_glinternet

Predict values for a penalized regression model with interactions.

Description

This function wraps the procedure for predicting values from a glinternet model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_glinternet'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_glinternet.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

predict_model.easy_glmnet

Predict values for a penalized regression model.

Description

This function wraps the procedure for predicting values from a glmnet model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_glmnet'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_glmnet.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

predict_model.easy_neural_network

Predict values for a neural network model.

Description

This function wraps the procedure for predicting values from a neural network model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_neural_network'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_neural_network.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

predict_model.easy_random_forest

Predict values for a random forest regression model.

Description

This function wraps the procedure for predicting values from a random forest model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_random_forest'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_random_forest.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

predict_model.easy_support_vector_machine

Predict values for a support vector machine regression model.

Description

This function wraps the procedure for predicting values from a support vector machine model and makes it accessible to the easyml core framework.

Usage

```
## S3 method for class 'easy_support_vector_machine'
predict_model(object, newx = NULL)
```

Arguments

object	A list of class easy_support_vector_machine.
newx	A data.frame, the new data to use for predictions.

Value

A vector, the predicted values using the new data.

preprocess_identity *Preprocess data by leaving it exactly the way it is.*

Description

This function is the same as the identity function. Anything passed to it is returned untouched.

Usage

```
preprocess_identity(.data, categorical_variables = NULL)
```

Arguments

.data	A data.frame; the data to be analyzed.
categorical_variables	A logical vector; each value TRUE indicates that column in the data.frame is a categorical variable. Defaults to NULL.

Value

A list, containing one or two data.frames.

See Also

Other preprocess: preprocess_scale

preprocess_scale *Preprocess data by scaling it.*

Description

This function takes either a data.frame or a list of data.frames. In the event of the first, this function takes the dataset and will scale each column that is not categorical such that that column has zero mean and unit variance. In the event of the second, this function takes the training dataset and will identify the parameters needed to scale the training dataset such that each column that is not categorical has zero mean and unit variance, and then will apply those parameters to each column in the testing dataset that is not categorical.

Usage

```
preprocess_scale(.data, categorical_variables = NULL)
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`categorical_variables`	A logical vector; each value TRUE indicates that column in the data.frame is a categorical variable. Defaults to NULL.

Value

A list, containing one or two data.frames.

See Also

Other preprocess: `preprocess_identity`

`process_coefficients` *Process coefficients.*

Description

Takes the coefficients returned by the generate_coefficients function and prepares the coefficients for plotting.

Usage

```
process_coefficients(coefficients, survival_rate_cutoff = 0.05)
```

Arguments

`coefficients`	A data.frame, the replicated coefficients.
`survival_rate_cutoff`	A numeric vector of length one; for `easy_glmnet`, specifies the minimal threshold (as a percentage) a coefficient must appear out of n_samples. Defaults to 0.05.

Value

A data.frame; the replicated coefficients processed for easy plotting.

See Also

Other utils: `correlation_test`, `process_variable_importances`, `reduce_cores`, `remove_variables`

process_variable_importances *Process variable importances.*

Description

Takes the variable importances returned by the generate_variable_importances function and prepares the variable importances for plotting.

Usage

```
process_variable_importances(variable_importances)
```

Arguments

variable_importances
: A data.frame, the replicated coefficients.

Value

A data.frame; the replicated variable importances processed for easy plotting.

See Also

Other utils: correlation_test, process_coefficients, reduce_cores, remove_variables

prostate *Prostate data.*

Description

Prostate data.

Usage

```
prostate
```

Format

Data frame with columns

lcavol log(cancer volume)

lweight log(prostate weight)

age age

lbph log(benign prostatic hyperplasia amount)

svi seminal vesicle invasion

lcp log(capsular penetration)

gleason Gleason score

pgg45 percentage Gleason scores 4 or 5

lpsa log(prostate specific antigen)

Source

Stamey, T.A., Kabalin, J.N., McNeal, J.E., Johnstone, I.M., Freiha, F., Redwine, E.A. and Yang, N. (1989) Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate: II. radical prostatectomy treated patients, Journal of Urology 141(5), 1076-1083.

See Also

Other data: `cocaine_dependence`

Examples

```
data("prostate", package = "easyml")
```

---

reduce_cores *Reduce number of cores.*

Description

This function takes the number of cores specified by the user and reduces it to the maximum number of cores supported by the computer.

Usage

```
reduce_cores(n_core, cpu_count = NULL)
```

Arguments

`n_core`	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1L.
`cpu_count`	An integer vector of length one; specifies the number of cores potentially available to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1L.

Value

An integer vector of length one; specifies the number of cores to use for this analysis.

See Also

Other utils: `correlation_test`, `process_coefficients`, `process_variable_importances`, `remove_variables`

remove_variables *Remove variables from a dataset.*

Description

This utility function removes variables from a data.frame.

Usage

```
remove_variables(.data = NULL, exclude_variables = NULL)
```

Arguments

.data	A data.frame; the data to be analyzed.
exclude_variables	A character vector; the variables from the data set to exclude. Defaults to NULL.

Value

A data.frame; the data to be analyzed.

See Also

Other utils: correlation_test, process_coefficients, process_variable_importances, reduce_cores

resample_fold_train_test_split *Sample with respect to an identification vector*

Description

This will sample the training and test sets so that case identifiers (e.g. subject ID's) are not shared across training and test sets.

Usage

```
resample_fold_train_test_split(X, y, train_size = 0.667, foldid = NULL,
  random_state = NULL)
```

Arguments

X	A data.frame, the data to be resampled.
y	A numeric vector with two classes, 0 and 1.
train_size	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
foldid	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.
random_state	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

Value

A boolean vector of length n_obs where TRUE represents that observation should be in the train set.

See Also

Other resample: `resample_simple_train_test_split`, `resample_stratified_class_train_test_split`, `resample_stratified_simple_train_test_split`

`resample_simple_train_test_split` *Train test split.*

Description

This will split the data into train and test.

Usage

```
resample_simple_train_test_split(X, y, train_size = 0.667, foldid = NULL,
  random_state = NULL)
```

Arguments

X	A data.frame, the data to be resampled.
y	A numeric vector with two classes, 0 and 1.
train_size	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
foldid	Not currently supported in this function.
random_state	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

Value

A boolean vector of length n_obs where TRUE represents that observation should be in the train set.

See Also

Other resample: resample_fold_train_test_split, resample_stratified_class_train_test_split, resample_stratified_simple_train_test_split

---

resample_stratified_class_train_test_split

Sample in equal proportion.

Description

This will sample in equal proportion.

Usage

```
resample_stratified_class_train_test_split(X, y, train_size = 0.667,
  foldid = NULL, random_state = NULL)
```

Arguments

X	A data.frame, the data to be resampled.
y	A numeric vector with two classes, 0 and 1.
train_size	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
foldid	Not currently supported in this function.
random_state	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

Value

A boolean vector of length n_obs where TRUE represents that observation should be in the train set.

See Also

Other resample: resample_fold_train_test_split, resample_simple_train_test_split, resample_stratified_si

resample_stratified_simple_train_test_split

Sample in equal proportion.

Description

This will sample in equal proportion.

Usage

```
resample_stratified_simple_train_test_split(X, y, train_size = 0.667,
  foldid = NULL, random_state = NULL)
```

Arguments

X	A data.frame, the data to be resampled.
y	A numeric vector with two classes, 0 and 1.
train_size	A numeric vector of length one; specifies what proportion of the data should be used for the training data set. Defaults to 0.667.
foldid	A vector with length equal to `length(y)` which identifies cases belonging to the same fold.
random_state	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

Value

A boolean vector of length n_obs where TRUE represents that observation should be in the train set.

See Also

Other resample: resample_fold_train_test_split, resample_simple_train_test_split, resample_stratified_cl

set_categorical_variables

Set categorical variables.

Description

This helper functions determines a logical boolean vector based on the column names and the designation for which ones are categorical variables.

Usage

```
set_categorical_variables(column_names, categorical_variables = NULL)
```

Arguments

column_names	A character vector; the column names of the data for this analysis.
categorical_variables	A character vector; the variables that are categorical. Defaults to NULL.

Value

NULL, or if categorical_variables is not NULL, then a logical vector of length length(column_names where TRUE represents that column is a categorical variable.

See Also

Other setters: set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

---

set_column_names *Set column names.*

Description

This functions helps decide what the updated column names of a data.frame should be within the easyml framework based on the dependent variable, preprocessing function, exclusionary variables, and categorical variables.

Usage

```
set_column_names(column_names, dependent_variable, preprocess = NULL,
  exclude_variables = NULL, categorical_variables = NULL)
```

Arguments

column_names	A character vector; the column names of the data for this analysis.
dependent_variable	A character vector of length one; the dependent variable for this analysis.
preprocess	A function; the function for preprocessing the data. Defaults to NULL.
exclude_variables	A character vector; the variables from the data set to exclude. Defaults to NULL.
categorical_variables	A character vector; the variables that are categorical. Defaults to NULL.

Value

The updated columns, in the correct order for preprocessing.

See Also

Other setters: set_categorical_variables, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

---

`set_cores` *Set cores.*

Description

Please note this affects global state and sets the number of cores by running `options(mc.cores = n_core)`.

Usage

```
set_cores(n_core)
```

Arguments

`n_core`	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1L.

Value

NULL.

See Also

Other setters: set_categorical_variables, set_column_names, set_dependent_variable, set_independent_variabl
set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

---

`set_dependent_variable` *Set dependent variable.*

Description

This helper functions isolates the dependent variable in a data.frame.

Usage

```
set_dependent_variable(.data, dependent_variable)
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.

Value

A vector, the dependent variable of the analysis.

See Also

Other setters: `set_categorical_variables`, `set_column_names`, `set_cores`, `set_independent_variables`, `set_looper_`, `set_looper`, `set_measure`, `set_parallel`, `set_plot_model_performance`, `set_plot_predictions`, `set_preprocess`, `set_random_state`, `set_resample`

`set_independent_variables` *Set independent variables.*

Description

This helper functions isolates the independent variables in a data.frame.

Usage

```
set_independent_variables(.data, dependent_variable)
```

Arguments

`.data`	A data.frame; the data to be analyzed.
`dependent_variable`	A character vector of length one; the dependent variable for this analysis.

Value

A data.frame, the independent variables of the analysis.

See Also

Other setters: `set_categorical_variables`, `set_column_names`, `set_cores`, `set_dependent_variable`, `set_looper_`, `set_looper`, `set_measure`, `set_parallel`, `set_plot_model_performance`, `set_plot_predictions`, `set_preprocess`, `set_random_state`, `set_resample`

set_looper *Set looper.*

Description

This function decides which looper (a functional like lapply) to run. Please note this affects global state and sets the number of cores by ultimately running `options(mc.cores = n_core)`.

Usage

```
set_looper(progress_bar = FALSE, n_core = 1)
```

Arguments

progress_bar	A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to TRUE.
n_core	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1L.

Value

The looper to use depending on progress bar and whether to run in parallel or not.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_measure, set_parallel, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

set_looper_ *Set looper.*

Description

This function decides which looper (a functional like lapply) to run. This function does not affect global state.

Usage

```
set_looper_(progress_bar = FALSE, parallel = FALSE)
```

Arguments

progress_bar	A logical vector of length one; specifies whether to display a progress bar during calculations. Defaults to FALSE.
parallel	A logical vector of length one; specifies whether to run calculations in parallel. Defaults to FALSE.

Value

The looper to use depending on progress bar and whether to run in parallel or not.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper, set_measure, set_parallel, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

---

set_measure *Set measure function.*

Description

Sets the function responsible for measuring the results.

Usage

```
set_measure(measure = NULL, algorithm, family)
```

Arguments

measure	A function; the function for measuring the results. Defaults to NULL.
algorithm	A character vector of length one; the algorithm to run on the data. Choices are one of c("glmnet", "random_forest", "support_vector_machine").
family	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".

Value

A function; the function for measuring the results.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_parallel, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

set_parallel *Set parallel.*

Description

This helper function decides whether the analysis should be run in parallel based on the number of cores specified.

Usage

```
set_parallel(n_core)
```

Arguments

n_core	An integer vector of length one; specifies the number of cores to use for this analysis. Currently only works on Mac OSx and Unix/Linux systems. Defaults to 1L.

Value

A logical vector of length one; whether analysis should be run in parallel or not.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_plot_model_performance, set_plot_predictions, set_preprocess, set_random_state, set_resample

set_plot_model_performance *Set plot model performance function.*

Description

Sets the function responsible for plotting the measures of model performance generated from the predictions generated from a fitted model.

Usage

```
set_plot_model_performance(measure)
```

Arguments

measure	A function; the function for measuring the results. Defaults to NULL.

Value

TA function; the function for plotting the measures of model performance generated from the predictions generated from a fitted model.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_predictions, set_preprocess, set_random_state, set_resample

---

`set_plot_predictions` *Set plot predictions function.*

Description

Sets the function responsible for plotting the predictions generated from a fitted model.

Usage

```
set_plot_predictions(algorithm, family)
```

Arguments

`algorithm`	A character vector of length one; the algorithm to run on the data. Choices are one of c("glmnet", "random_forest", "support_vector_machine").
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".

Value

A function; the function for plotting the predictions generated from a fitted model.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance, set_preprocess, set_random_state, set_resample

set_preprocess *Set preprocess function.*

Description

Sets the function responsible for preprocessing the data.

Usage

```
set_preprocess(preprocess = NULL, algorithm)
```

Arguments

preprocess	A function; the function for preprocessing the data. Defaults to NULL.
algorithm	A character vector of length one; the algorithm to run on the data. Choices are one of c("glmnet", "random_forest", "support_vector_machine").

Value

A function; the function for preprocessing the data.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance set_plot_predictions, set_random_state, set_resample

set_random_state *Set random state.*

Description

Sets the random state to a specific seed. Please note this function affects global state.

Usage

```
set_random_state(random_state = NULL)
```

Arguments

random_state	An integer vector of length one; specifies the seed to be used for the analysis. Defaults to NULL.

Value

NULL.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance set_plot_predictions, set_preprocess, set_resample

---

`set_resample` *Set resample function.*

Description

Sets the function responsible for resampling the data.

Usage

```
set_resample(resample = NULL, family = NULL)
```

Arguments

`resample`	A function; the function for resampling the data. Defaults to NULL.
`family`	A character vector of length one; the type of regression to run on the data. Choices are one of c("gaussian", "binomial"). Defaults to "gaussian".

Value

A function; the function for resampling the data.

See Also

Other setters: set_categorical_variables, set_column_names, set_cores, set_dependent_variable, set_independent_variables, set_looper_, set_looper, set_measure, set_parallel, set_plot_model_performance set_plot_predictions, set_preprocess, set_random_state

Index