Package ‘mikropml’

December 3, 2020

Title User-Friendly R Package for Supervised Machine Learning Pipelines
Version 0.0.2
Date 2020-12-02
BugReports https://github.com/SchlossLab/mikropml/issues
Description An interface to build machine learning models for classification and regression problems. ‘mikropml’ implements the ML pipeline described by Topçuo˘glu et al. (2020) ⟨doi:10.1128/mBio.00434-20⟩ with reasonable default options for data preprocessing, hyperparameter tuning, cross-validation, testing, model evaluation, and interpretation steps. See the website ⟨http://www.schlosslab.org/mikropml⟩ for more information, documentation, and examples.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Imports caret, dplyr, e1071, glmnet, kernlab, MLmetrics, randomForest, rlang, rpart, stats, utils, xgboost
Suggests doFuture, foreach, future, future.apply, ggplot2, knitr, purrr, rmarkdown, testthat, tidyr
VignetteBuilder knitr
Depends R (>= 2.10)
NeedsCompilation no
Maintainer  Kelly Sovacool <sovacool@umich.edu>
Repository CRAN
Date/Publication  2020-12-03 08:10:02 UTC

R topics documented:
calc_perf_metrics  .................................................. 3
combine_hp_performance ........................................ 4
define_cv ................................................................. 5
get_caret_processed_df ........................................ 6
get_corr_feats ...................................................... 7
get_feature_importance ........................................... 8
get_hp_performance ................................................ 9
get_hyperparams_list .......................................... 10
get_outcome_type ................................................ 11
get_partition_indices ........................................... 12
get_performance_tbl ............................................. 13
get_perf_metric_fn .............................................. 14
get_perf_metric_name .......................................... 15
get_tuning_grid .................................................... 15
mikropml ............................................................... 16
otu_mini_bin ......................................................... 17
otu_mini_bin_results_glmnet ................................ 17
otu_mini_bin_results_rf ....................................... 18
otu_mini_bin_results_rpart2 .................................. 18
otu_mini_bin_results_svmRadial .............................. 18
otu_mini_bin_results_xgbTree .................................. 19
otu_mini_cont_results_glmnet .................................. 19
otu_mini_cv ........................................................... 19
otu_mini_multi ....................................................... 20
otu_mini_multi_results_glmnet ............................... 20
otu_small ............................................................. 21
plot_hp_performance ............................................. 21
plot_model_performance ....................................... 22
preprocess_data .................................................. 23
randomize_feature_order ..................................... 24
run_ml ................................................................. 25
tidy_perf_data ..................................................... 28
train_model .......................................................... 29

Index  31
calc_perf_metrics

Get performance metrics for test data

Description
Get performance metrics for test data

Usage

```
calc_perf_metrics(
  test_data,
  trained_model,
  outcome_colname,
  perf_metric_function,
  class_probs
)
```

Arguments

test_data
Held out test data: dataframe of outcome and features.

trained_model
Trained model from `caret::train()`.

outcome_colname
Column name as a string of the outcome variable (default NULL; the first column will be chosen automatically).

perf_metric_function
Function to calculate the performance metric to be used for cross-validation and test performance. Some functions are provided by caret (see `caret::defaultSummary()`). Defaults: binary classification = `twoClassSummary`, multi-class classification = `multiClassSummary`, regression = `defaultSummary`.

class_probs
Whether to use class probabilities (TRUE for categorical outcomes, FALSE for numeric outcomes).

Value
Dataframe of performance metrics.

Author(s)
Zena Lapp, <zenalapp@umich.edu>

Examples

```
results <- run_ml(otu_small, "glmnet", kfold = 2, cv_times = 2)
calc_perf_metrics(results$test_data,
  results$trained_model,
  "dx",
)
combine_hp_performance

Combine hyperparameter performance metrics for multiple train/test splits

Description

Combine hyperparameter performance metrics for multiple train/test splits generated by, for instance, looping in R or using a snakemake workflow on a high-performance computer.

Usage

combine_hp_performance(trained_model_lst)

Arguments

trained_model_lst

List of trained models.

Value

Named list:

- dat: Dataframe of performance metric for each group of hyperparameters
- params: Hyperparameters tuned.
- Metric: Performance metric used.

Author(s)

Zena Lapp, <zenalapp@umich.edu>

Examples

results <- lapply(seq(100, 102), function(seed) {
  run_ml(otu_small, "glmnet", seed = seed, cv_times = 2, kfold = 2)
})
models <- lapply(results, function(x) x$trained_model)
combine_hp_performance(models)
Define cross-validation scheme and training parameters

**Description**

Define cross-validation scheme and training parameters

**Usage**

```r
define_cv(
  train_data,
  outcome_colname,
  hyperparams_list,
  perf_metric_function,
  class_probs,
  kfold = 5,
  cv_times = 100,
  groups = NULL
)
```

**Arguments**

- `train_data`: Dataframe for training model.
- `outcome_colname`: Column name as a string of the outcome variable (default NULL; the first column will be chosen automatically).
- `hyperparams_list`: Named list of lists of hyperparameters.
- `perf_metric_function`: Function to calculate the performance metric to be used for cross-validation and test performance. Some functions are provided by caret (see `caret::defaultSummary()`). Defaults: binary classification = `twoClassSummary`, multi-class classification = `multiClassSummary`, regression = `defaultSummary`.
- `class_probs`: Whether to use class probabilities (TRUE for categorical outcomes, FALSE for numeric outcomes).
- `kfold`: Fold number for k-fold cross-validation (default: 5).
- `cv_times`: Number of cross-validation partitions to create (default: 100).
- `groups`: Vector of groups to keep together when splitting the data into train and test sets, and for cross-validation. length matches the number of rows in the dataset (default: NULL).

**Value**

Caret object for trainControl that controls cross-validation
**get_caret_processed_df**

Get preprocessed dataframe for continuous variables

**Description**

Get preprocessed dataframe for continuous variables

**Usage**

```r
get_caret_processed_df(features, method)
```

**Arguments**

- **features**  
  Dataframe of features for machine learning
- **method**  
  Methods to preprocess the data, described in `caret::preProcess()` (default: c("center","scale"), use NULL for no normalization).

**Value**

Named list:

- **processed**: Dataframe of processed features.
- **removed**: Names of any features removed during preprocessing.

**Author(s)**

Zena Lapp, <zenalapp@umich.edu>
**get_corr_feats**

*Identify correlated features*

**Description**

Identify correlated features

**Usage**

    get_corr_feats(features, corr_thresh = 1, group_neg_corr = TRUE)

**Arguments**

- **features**: Features used for machine learning.
- **corr_thresh**: For feature importance, group correlations above or equal to corr_thres (range 0 to 1; default: 1).
- **group_neg_corr**: Whether to group negatively correlated features together (e.g. c(0,1) and c(1,0)).

**Value**

Dataframe of correlated features where the columns are feature1, feature2, and the correlation between those two features (anything exceeding corr_thres).

**Author(s)**

Begüm Topçuoğlu, <topcuoglu.begum@gmail.com>

Zena Lapp, <zenalapp@umich.edu>

**Examples**

    set.seed(0)
    mat <- matrix(runif(100), nrow = 20)
    rownames(mat) <- 1:nrow(mat)
    colnames(mat) <- 1:ncol(mat)
    get_corr_feats(mat, 0.4)
get_feature_importance

*Get feature importance using permutation method*

**Description**

Calculates feature importance using a trained model and test data. Requires the `future.apply` package.

**Usage**

```r
get_feature_importance(
  trained_model,
  train_data,
  test_data,
  outcome_colname,
  perf_metric_function,
  perf_metric_name,
  class_probs,
  method,
  seed = NA,
  corr_thresh = 1
)
```

**Arguments**

- **trained_model**: Trained model from `caret::train()`.
- **train_data**: Training data: dataframe of outcome and features.
- **test_data**: Held out test data: dataframe of outcome and features.
- **outcome_colname**: Column name as a string of the outcome variable (default `NULL`; the first column will be chosen automatically).
- **perf_metric_function**: Function to calculate the performance metric to be used for cross-validation and test performance. Some functions are provided by caret (see `caret::defaultSummary()`). Defaults: binary classification = `twoClassSummary`, multi-class classification = `multiClassSummary`, regression = `defaultSummary`.
- **perf_metric_name**: The column name from the output of the function provided to `perf_metric_function` that is to be used as the performance metric. Defaults: binary classification = "ROC", multi-class classification = "logLoss", regression = "RMSE".
- **class_probs**: Whether to use class probabilities (TRUE for categorical outcomes, FALSE for numeric outcomes).
- **method**: ML method. Options: c("glmnet","rf","rpart2","svmRadial","xgbTree").
  - `glmnet`: linear, logistic, or multiclass regression
• rf: random forest
• rpart2: decision tree
• svmRadial: support vector machine
• xgbTree: xgboost

seed Random seed (default: NA). Your results will only be reproducible if you set a seed.
corr_thresh For feature importance, group correlations above or equal to corr_thres (range 0 to 1; default: 1).

Value

Dataframe with performance metrics for when each feature (or group of correlated features; names) is permuted (perf_metric), and differences between test performance metric and permuted performance metric (perf_metric_diff). The performance metric name (perf_metric_name) and seed (seed) are also returned.

Author(s)

Begüm Topçuoğlu, <topcuoglu.begum@gmail.com>
Zena Lapp, <zenalapp@umich.edu>

Examples

results <- run_ml(otu_small, "glmnet", kfold = 2, cv_times = 2)
names(results$trained_model$trainingData)[1] <- "dx"
get_feature_importance(results$trained_model, 
  results$trained_model$trainingData, results$test_data, 
  "dx", 
  multiClassSummary, "AUC", 
  class_probs = TRUE, method = "glmnet"
)
get_hyperparams_list

Arguments

trained_model trained model (e.g. from run_ml())

Value

Named list:

- dat: Dataframe of performance metric for each group of hyperparameters.
- params: Hyperparameters tuned.
- metric: Performance metric used.

Author(s)

Zena Lapp, <zenalapp@umich.edu>
Kelly Sovacool <sovacool@umich.edu>

Examples

get_hp_performance(otu_mini_bin_results_glmnet$trained_model)

get_hyperparams_list dataset, method

Arguments

dataset Dataframe with an outcome variable and other columns as features.
method ML method. Options: c("glmnet","rf","rpart2","svmRadial","xgbTree").

- glmnet: linear, logistic, or multiclass regression
- rf: random forest
- rpart2: decision tree
- svmRadial: support vector machine
- xgbTree: xgboost

Value

Named list of hyperparameters.

Description

For more details see the vignette on hyperparameter tuning.

Usage

get_hyperparams_list(dataset, method)
get_outcome_type

Author(s)

Kelly Sovacool, <sovacool@umich.edu>

Examples

get_hyperparams_list(otu_mini_bin, "rf")
get_hyperparams_list(otu_small, "rf")
get_hyperparams_list(otu_mini_bin, "rpart2")
get_hyperparams_list(otu_small, "rpart2")

get_outcome_type(outcomes_vec)

Description

If the outcome is numeric, the type is continuous. Otherwise, the outcome type is binary if there are only two outcomes or multiclass if there are more than two outcomes.

Usage

get_outcome_type(outcomes_vec)

Arguments

outcomes_vec Vector of outcomes.

Value

Outcome type (continuous, binary, or multiclass).

Author(s)

Zena Lapp, <zenalapp@umich.edu>

Examples

get_outcome_type(c(1, 2, 1))
get_outcome_type(c("a", "b", "b"))
get_outcome_type(c("a", "b", "c"))
get_partition_indices  Select indices to partition the data into training & testing sets.

Description

Use this function to get the row indices for the training set.

Usage

get_partition_indices(outcomes, training_frac = 0.8, groups = NULL)

Arguments

outcomes vector of outcomes
training_frac max fraction of data for the training set (default: 0.8)
groups vector of groups. length must match the number of rows in the dataset. (default: NULL)

Details

If groups is NULL, uses createDataPartition. Otherwise, uses create_grouped_data_partition().

Set the seed prior to calling this function if you would like your data partitions to be reproducible (recommended).

Value

Vector of row indices for the training set.

Author(s)

Kelly Sovacool, sovacool@umich.edu

Examples

training inds <- get_partition_indices(otu_mini_bin$dx)
train_data <- otu_mini_bin[training inds, ]
test_data <- otu_mini_bin[-training inds, ]
**get_performance_tbl**

Get model performance metrics as a one-row tibble

### Description
Get model performance metrics as a one-row tibble

### Usage

```r
get_performance_tbl(
  trained_model, 
  test_data, 
  outcome_colname, 
  perf_metric_function, 
  perf_metric_name, 
  class_probs, 
  method, 
  seed = NA
)
```

### Arguments

- **trained_model**  Trained model from `caret::train()`.  
- **test_data**  Held out test data: dataframe of outcome and features.  
- **outcome_colname**  Column name as a string of the outcome variable (default NULL; the first column will be chosen automatically).  
- **perf_metric_function**  Function to calculate the performance metric to be used for cross-validation and test performance. Some functions are provided by caret (see `caret::defaultSummary()`). Defaults: binary classification = `twoClassSummary`, multi-class classification = `multiClassSummary`, regression = `defaultSummary`.  
- **perf_metric_name**  The column name from the output of the function provided to `perf_metric_function` that is to be used as the performance metric. Defaults: binary classification = "ROC", multi-class classification = "logLoss", regression = "RMSE".  
- **class_probs**  Whether to use class probabilities (TRUE for categorical outcomes, FALSE for numeric outcomes).  
- **method**  ML method. Options: c("glmnet","rf","rpart2","svmRadial","xgbTree").  
  - glmnet: linear, logistic, or multiclass regression  
  - rf: random forest  
  - rpart2: decision tree  
  - svmRadial: support vector machine  
  - xgbTree: xgboost  
- **seed**  Random seed (default: NA). Your results will only be reproducible if you set a seed.
get_perf_metric_fn

Value

A one-row tibble with columns cv_auroc, column for each of the performance metrics for the test data method, and seed.

Author(s)

Kelly Sovacool, <sovacool@umich.edu>
Zena Lapp, <zenalapp@umich.edu>

Examples

results <- run_ml(otu_small, "glmnet", kfold = 2, cv_times = 2)
names(results$trained_model$trainingData)[1] <- "dx"
get_performance_tbl(results$trained_model, results$test_data, "dx",
  multiClassSummary, "AUC",
  class_probs = TRUE,
  method = "glmnet"
)

get_perf_metric_fn

Get default performance metric function

Description

Get default performance metric function

Usage

get_perf_metric_fn(outcome_type)

Arguments

outcome_type Type of outcome (one of: "continuous","binary","multiclass").

Value

Performance metric function.

Author(s)

Zena Lapp, <zenalapp@umich.edu>
### get_perf_metric_name

**Get default performance metric name**

**Description**

Get default performance metric name for cross-validation.

**Usage**

```python
get_perf_metric_name(outcome_type)
```

**Arguments**

- `outcome_type`  
  Type of outcome (one of: "continuous","binary","multiclass").

**Value**

Performance metric name.

**Author(s)**

Zena Lapp, <zenalapp@umich.edu>

**Examples**

```python
get_perf_metric_name("continuous")
get_perf_metric_name("binary")
get_perf_metric_name("multiclass")
```

---

### get_tuning_grid

**Generate the tuning grid for tuning hyperparameters**

**Description**

Generate the tuning grid for tuning hyperparameters.

**Usage**

```python
get_tuning_grid(hyperparams_list, method)
```
Arguments

hyperparams_list
Named list of lists of hyperparameters.

method ML method. Options: c("glmnet","rf","rpart2","svmRadial","xgbTree").
  • glmnet: linear, logistic, or multiclass regression
  • rf: random forest
  • rpart2: decision tree
  • svmRadial: support vector machine
  • xgbTree: xgboost

Value
The tuning grid.

Author(s)
Begüm Topçuo˘glu, <topcuoglu.begum@gmail.com>
Kelly Sovacool, <sovacool@umich.edu>

Examples

ml_method <- "glmnet"
hpars_list <- get_hyperparams_list(otu_small, ml_method)
get_tuning_grid(hpars_list, ml_method)

mikropml mikropml: User-Friendly R Package for Robust Machine Learning Pipelines

Description
mikropml implements robust machine learning pipelines using regression, support vector machines, decision trees, random forest, or gradient-boosted trees. The main functions are prepocess_data() to process your data prior to running machine learning, and run_ml() to run machine learning.

Authors
• Begüm D. Topçuoğlu (ORCID)
• Zena Lapp (ORCID)
• Kelly L. Sovacool (ORCID)
• Evan Snitkin (ORCID)
• Jenna Wiens (ORCID)
• Patrick D. Schloss (ORCID)
See vignettes

- Introduction
- Preprocessing data
- Hyperparameter tuning
- Parallel processing
- The mikropml paper

---

**otu_mini_bin**  
*Mini OTU abundance dataset*

**Description**

A dataset containing relatives abundances of OTUs for human stool samples with a binary outcome, `dx`. This is a subset of `otu_small`.

**Usage**

`otu_mini_bin`

**Format**

A data frame. The `dx` column is the diagnosis: healthy or cancerous (colorectal). All other columns are OTU relative abundances.

---

**otu_mini_bin_results_glmnet**

*Results from running the pipeline with L2 logistic regression on otu_mini_bin with feature importance and grouping*

**Description**

Results from running the pipeline with L2 logistic regression on `otu_mini_bin` with feature importance and grouping.

**Usage**

`otu_mini_bin_results_glmnet`

**Format**

An object of class `list` of length 4.
otu_mini_bin_results_rf

Results from running the pipeline with random forest on otu_mini_bin

Description

Results from running the pipeline with random forest on otu_mini_bin

Usage

otu_mini_bin_results_rf

Format

An object of class list of length 4.

otu_mini_bin_results_rpart2

Results from running the pipeline with rpart2 on otu_mini_bin

Description

Results from running the pipeline with rpart2 on otu_mini_bin

Usage

otu_mini_bin_results_rpart2

Format

An object of class list of length 4.

otu_mini_bin_results_svmRadial

Results from running the pipeline with svmRadial on otu_mini_bin

Description

Results from running the pipeline with svmRadial on otu_mini_bin

Usage

otu_mini_bin_results_svmRadial

Format

An object of class list of length 4.
**otu_mini_bin_results_xgbTree**

Results from running the pipeline with xgbTree on otu_mini_bin

**Description**

Results from running the pipeline with xgbTree on otu_mini_bin

**Usage**

```
otu_mini_bin_results_xgbTree
```

**Format**

An object of class list of length 4.

---

**otu_mini_cont_results_glmnet**

Results from running the pipeline with glmnet on otu_mini_bin with Otu00001 as the outcome

**Description**

Results from running the pipeline with glmnet on otu_mini_bin with Otu00001 as the outcome

**Usage**

```
otu_mini_cont_results_glmnet
```

**Format**

An object of class list of length 4.

---

**otu_mini_cv**

Cross validation on train_data_mini with grouped features.

**Description**

Cross validation on train_data_mini with grouped features.

**Usage**

```
otu_mini_cv
```

**Format**

An object of class list of length 27.
**otu_mini_multi**  
*Mini OTU abundance dataset with 3 categorical variables*

**Description**
A dataset containing relatives abundances of OTUs for human stool samples

**Usage**

`otu_mini_multi`

**Format**
A data frame The *dx* column is the colorectal cancer diagnosis: adenoma, carcinoma, normal. All other columns are OTU relative abundances.

---

**otu_mini_multi_results_glmnet**  
*Results from running the pipeline with glmnet on otu_mini_multi for multiclass outcomes*

**Description**
Results from running the pipeline with glmnet on otu_mini_multi for multiclass outcomes

**Usage**

`otu_mini_multi_results_glmnet`

**Format**
An object of class list of length 4.
otu_small

Small OTU abundance dataset

Description
A dataset containing relatives abundances of 60 OTUs for 60 human stool samples. This is a subset of the data provided in extdata/otu_large.csv, which was used in Topçuoğlu et al. 2020.

Usage
otu_small

Format
A data frame with 60 rows and 61 variables. The dx column is the diagnosis: healthy or cancerous (colorectal). All other columns are OTU relative abundances.

plot_hp_performance
Plot hyperparameter performance metrics

Description
Plot hyperparameter performance metrics

Usage
plot_hp_performance(dat, param_col, metric_col)

Arguments

dat
dataframe of hyperparameters and performance metric (e.g. from get_hp_performance() or combine_hp_performance())

param_col
hyperparameter to be plotted. must be a column in dat.

metric_col
performance metric. must be a column in dat.

Value
ggplot of hyperparameter performance.

Author(s)
Zena Lapp, <zenalapp@umich.edu>
Kelly Sovacool <sovacool@umich.edu>
Examples

# plot for a single `run_ml()` call
hp_metrics <- get_hp_performance(otu_mini_bin_results_glmnet$trained_model)
hp_metrics
plot_hp_performance(hp_metrics$dat, lambda, AUC)

# plot for multiple `run_ml()` calls
results <- lapply(seq(100, 102), function(seed) {
  run_ml(otu_small, "glmnet", seed = seed)
})
models <- lapply(results, function(x) x$trained_model)
hp_metrics <- combine_hp_performance(models)
plot_hp_performance(hp_metrics$dat, lambda, AUC)
```
perf_df <- lapply(results_lst, function(result) {
  result[["performance"]]
}) %>%
  dplyr::bind_rows()
# plot the performance results
p <- plot_model_performance(perf_df)

# call `run_ml()` with different ML methods
param_grid <- expand.grid(
  seeds = seq(100, 104),
  methods = c("glmnet", "rf")
)
results_mtx <- mapply(
  function(seed, method) {
    run_ml(otu_mini_bin, method, seed = seed, kfold = 2)
  },
  param_grid$seeds, param_grid$methods
)
# extract and combine the performance results
perf_df2 <- dplyr::bind_rows(results_mtx["performance", ])  
# plot the performance results
p <- plot_model_performance(perf_df2)

# you can continue adding layers to customize the plot
p +
  theme_classic() +
  scale_color_brewer(palette = "Dark2") +
  coord_flip()
```

---

**preprocess_data**  
Preprocess data prior to running machine learning

**Description**

Function to preprocess your data for input into `run_ml()`.

**Usage**

```r
preprocess_data(
  dataset,
  outcome_colname,
  method = c("center", "scale"),
  remove_var = "nzv",
  collapse_corr_feats = TRUE,
  to_numeric = TRUE,
  group_neg_corr = TRUE
)
```
randomize_feature_order

Arguments

dataset          Dataframe with an outcome variable and other columns as features.
outcome_colname  Column name as a string of the outcome variable (default NULL; the first column
                 will be chosen automatically).
method           Methods to preprocess the data, described in caret::preProcess() (default: c("center", "scale"), use NULL for no normalization).
remove_var       Whether to remove variables with near-zero variance (‘nzv’; default), zero variance (‘zv’), or none (NULL).
collapse_corr_feats   Whether to keep only one of perfectly correlated features.
to_numeric       Whether to change features to numeric where possible.
group_neg_corr   Whether to group negatively correlated features together (e.g. c(0,1) and c(1,0)).

Value

Named list including:

- dat_transformed: Preprocessed data.
- grp_feats: If features were grouped together, a named list of the features corresponding to each group.
- removed_feats: Any features that were removed during preprocessing (e.g. because there was zero variance or near-zero variance for those features).

More details

See the preprocessing vignette for more details.

Author(s)

Zena Lapp, <zenalapp@umich.edu>

Examples

preprocess_data(mikropml::otu_small, "dx")

Description

Randomize feature order to eliminate any position-dependent effects
**run_ml**

**Usage**

```r
randomize_feature_order(dataset, outcome_colname)
```

**Arguments**

- `dataset` Dataframe with an outcome variable and other columns as features.
- `outcome_colname` Column name as a string of the outcome variable (default `NULL`; the first column will be chosen automatically).

**Value**

Dataset with feature order randomized.

**Author(s)**

Nick Lesniak, <nlesniak@umich.edu>
Kelly Sovacool, <sovacool@umich.edu>

**Examples**

```r
dat <- data.frame(
  outcome = c("1", "2", "3"),
  a = 4:6, b = 7:9, c = 10:12, d = 13:15
)
randomize_feature_order(dat, "outcome")
```

---

**Description**

This function runs machine learning (ML), evaluates the best model, and optionally calculates feature importance using a robust framework outlined in Topçuoglu et al. 2020 (doi:10.1128/mBio.00434-20). Required inputs are a dataframe with an outcome variable and other columns as features, as well as the ML method. See `vignette('introduction')` for more details.

**Usage**

```r
run_ml(
  dataset,
  method,
  outcome_colname = NULL,
  hyperparameters = NULL,
  find_feature_importance = FALSE,
  kfold = 5,
  cv_times = 100,
)```
training_frac = 0.8,
perf_metric_function = NULL,
perf_metric_name = NULL,
groups = NULL,
corr_thresh = 1,
ntree = 1000,
seed = NA
)

**Arguments**

dataset  
Dataframe with an outcome variable and other columns as features.

method  
ML method. Options: c("glmnet","rf","rpart2","svmRadial","xgbTree").
- glmnet: linear, logistic, or multiclass regression
- rf: random forest
- rpart2: decision tree
- svmRadial: support vector machine
- xgbTree: xgboost

outcome_colname  
Column name as a string of the outcome variable (default NULL; the first column will be chosen automatically).

hyperparameters
Dataframe of hyperparameters (default NULL; sensible defaults will be chosen automatically).

find_feature_importance
Run permutation importance (default: FALSE). TRUE is recommended if you would like to identify features important for predicting your outcome, but it is resource-intensive.

kfold
Fold number for k-fold cross-validation (default: 5).

cv_times
Number of cross-validation partitions to create (default: 100).

training_frac
Fraction of data for training set (default: 0.8). The remaining data will be used in the testing set.

perf_metric_function
Function to calculate the performance metric to be used for cross-validation and test performance. Some functions are provided by caret (see caret::defaultSummary()). Defaults: binary classification = twoClassSummary, multi-class classification = multiClassSummary, regression = defaultSummary.

perf_metric_name
The column name from the output of the function provided to perf_metric_function that is to be used as the performance metric. Defaults: binary classification = "ROC", multi-class classification = "logLoss", regression = "RMSE".

groups
Vector of groups to keep together when splitting the data into train and test sets, and for cross-validation. length matches the number of rows in the dataset (default: NULL).
For feature importance, group correlations above or equal to `corr_thresh` (range 0 to 1; default: 1).

For random forest, how many trees to use (default: 1000). Note that caret doesn’t allow this parameter to be tuned.

Random seed (default: NA). Your results will only be reproducible if you set a seed.

**Value**

Named list with results:

- trained_model: Output of `caret::train()`, including the best model.
- test_data: Part of the data that was used for testing.
- performance: Dataframe of performance metrics. The first column is the cross-validation performance metric, and the last two columns are the ML method used and the seed (if one was set), respectively. All other columns are performance metrics calculated on the test data. This contains only one row, so you can easily combine performance dataframes from multiple calls to `run_ml()` (see vignette("parallel")).
- feature_importance: If feature importances were calculated, a dataframe where each row is a feature or correlated group. The columns are the performance metric of the permuted data, the difference between the true performance metric and the performance metric of the permuted data (true - permuted), the feature name, the ML method, the performance metric name, and the seed (if provided). For AUC and RMSE, the higher perf_metric_diff is, the more important that feature is for predicting the outcome. For log loss, the lower perf_metric_diff is, the more important that feature is for predicting the outcome.

**More details**

For more details, please see the vignettes.

**Author(s)**

Begüm Topçuoğlu, <topcuoglu.begum@gmail.com>
Zena Lapp, <zenalapp@umich.edu>
Kelly Sovacool, <sovacool@umich.edu>

**Examples**

```r
run_ml(otu_small, "glmnet", 
  seed = 2019 
)
run_ml(otu_small, "rf", 
  outcome_colname = "dx", 
  find_feature_importance = TRUE 
)
```
tidy_perf_data  
Tidy the performance dataframe

Description

Used by plot_model_performance().

Usage

tidy_perf_data(performance_df)

Arguments

performance_df  dataframe of performance results from multiple calls to run_ml()

Value

Tidy dataframe with model performance metrics.

Author(s)

Begüm Topçuoglu, <topcuoglu.begum@gmail.com>
Kelly Sovacool, <sovacool@umich.edu>

Examples

# call `run_ml()` multiple times with different seeds
results_lst <- lapply(seq(100, 104), function(seed) {
  run_ml(otu_small, "glmnet", seed = seed)
})
# extract and combine the performance results
perf_df <- lapply(results_lst, function(result) {
  result[["performance"]]
}) %>%
  dplyr::bind_rows()
# make it pretty!
tidy_perf_data(perf_df)
Description
Train model using `caret::train()`.

Usage
```r
train_model(
  model_formula,
  train_data,
  method,
  cv,
  perf_metric_name,
  tune_grid,
  ntree
)
```

Arguments

- `model_formula`: Model formula.
- `train_data`: Training data.
- `method`: ML method. Options: c("glmnet","rf","rpart2","svmRadial","xgbTree").
  - `glmnet`: linear, logistic, or multiclass regression
  - `rf`: random forest
  - `rpart2`: decision tree
  - `svmRadial`: support vector machine
  - `xgbTree`: xgboost
- `cv`: Cross-validation caret scheme.
- `perf_metric_name`: The column name from the output of the function provided to `perf_metric_function` that is to be used as the performance metric. Defaults: binary classification = "ROC", multi-class classification = "logLoss", regression = "RMSE".
- `tune_grid`: Tuning grid.
- `ntree`: For random forest, how many trees to use (default: 1000). Note that caret doesn’t allow this parameter to be tuned.

Details
TODO: Add example.

Value
Trained model from `caret::train()`.
Author(s)

Zena Lapp, <zenalapp@umich.edu>
Index

* datasets
  otu_mini_bin, 17
  otu_mini_bin_results_glmnet, 17
  otu_mini_bin_results_rf, 18
  otu_mini_bin_results_rpart2, 18
  otu_mini_bin_results_svmRadial, 18
  otu_mini_bin_results_xgbTree, 19
  otu_mini_cont_results_glmnet, 19
  otu_mini_cv, 19
  otu_mini_multi, 20
  otu_mini_multi_results_glmnet, 20
  otu_small, 21

  calc_perf_metrics, 3
  caret::defaultSummary(), 3, 5, 8, 13, 26
  caret::preProcess(), 6, 24
  caret::train(), 3, 8, 13, 27, 29
  combine_hp_performance, 4
  createDataPartition, 12

  define_cv, 5

  get_caret_processed_df, 6
  get_corr_feats, 7
  get_feature_importance, 8
  get_hp_performance, 9
  get_hyperparams_list, 10
  get_outcome_type, 11
  get_partition_indices, 12
  get_perf_metric_fn, 14
  get_perf_metric_name, 15
  get_performance_tbl, 13
  get_tuning_grid, 15

  mikropml, 16

  otu_mini_bin, 17
  otu_mini_bin_results_glmnet, 17
  otu_mini_bin_results_rf, 18
  otu_mini_bin_results_rpart2, 18
  otu_mini_bin_results_svmRadial, 18
  otu_mini_cont_results_glmnet, 19
  otu_mini_cv, 19
  otu_mini_multi, 20
  otu_mini_multi_results_glmnet, 20
  otu_small, 21

  plot_hp_performance, 21
  plot_model_performance, 22
  preprocess_data, 23

  randomize_feature_order, 24
  run_ml, 25
  run_ml(), 23

  tidy_perf_data, 28
  train_model, 29