Package: pheble (via r-universe)

September 18, 2024

Title Classifying High-Dimensional Phenotypes with Ensemble Learning

Version 0.1.0

Description A system for binary and multi-class classification of high-dimensional phenotypic data using ensemble learning. By combining predictions from different classification models, this package attempts to improve performance over individual learners. The pre-processing, training, validation, and testing are performed end-to-end to minimize user input and simplify the process of classification.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports adabag, base, C50, caret, caTools, data.table, doParallel, dplyr, e1071, earth, evtree, frbs, glmnet, gmodels, hda, HDclassif, ipred, kernlab, kknn, klaR, magrittr, MASS, Matrix, mda, MLmetrics, nnet, parallel, party, pls, randomForest, rpartScore, sparseLDA, stats, themis, utils

Depends R (>= 2.10)

LazyData true

Suggests h2o

Repository https://jaydevine.r-universe.dev

RemoteUrl https://github.com/jaydevine/pheble

RemoteRef HEAD

RemoteSha 174e2a2ee661d5a4f4f0084aff0dfa38290c92bf

Contents

`Global variables.` *Global variables.*

Description

Global variables.

`ph_anomaly` *Detect anomalies.*

Description

The `ph_anomaly` function detects and removes anomalies with an autoencoder. Because it is general purpose, it can be applied to a variety of data types. The parameters in this function (e.g., activation, hidden, dropout_ratio) can be supplied as lists or vectors (see parameter details) to perform a grid search for the optimal hyperparameter combination. The autoencoder with the lowest reconstruction error is selected as the best model.

Usage

```
ph_anomaly(
  df,
  ids_col,
  class_col,
  method = "ae",
  scale = FALSE,
  center = NULL,
  sd = NULL,
  max_mem_size = "15g",
  port = 54321,
  train_seed = 123,
  hyper_params = list(),
  search = "random",
  tune_length = 100
)
```

Arguments

`df` A `data.frame` containing a column of ids, a column of classes, and an arbitrary number of predictors.

`ids_col` A `character` value for the name of the ids column.

`class_col` A `character` value for the name of the class column.

`method` A `character` value for the anomaly detection method: "ae" (default), "iso" (abbv. for extended isolation forest).

`scale` A `logical` value for whether to scale the data: FALSE (default). Recommended if `method = "ae"` and if user intends to train other models.

`center` Either a `logical` value or numeric-alike vector of length equal to the number of columns of data to scale in `df`, where ‘numeric-alike’ means that as.numeric(.) will be applied successfully if is.numeric(.) is not true: NULL (default). If `scale = TRUE`, this is set to `TRUE` and is used to subtract the mean.

`sd` Either a `logical` value or a numeric-alike vector of length equal to the number of columns of data to scale in `df`: NULL (default). If `scale = TRUE`, this is set to `TRUE` and is used to divide by the standard deviation.

`max_mem_size` A `character` value for the memory of an h2o session: "15g" (default).

`port` A `numeric` value for the port number of the H2O server.

`train_seed` A `numeric` value to set the control the randomness of creating resamples: 123 (default).

`hyper_params` A `list` of hyperparameters to perform a grid search.

- If `method = "ae"`, the "default" list is: list(missing_values_handling = "Skip", activation = c("Rectifier", "Tanh"), hidden = list(5, 25, 50, 100, 250, 500, nrow(df_h2o)), input_dropout_ratio = c(0, 0.1, 0.2, 0.3), rate = c(0, 0.01, 0.005, 0.001))
- If `method = "iso"`, the "default" list is: list(ntrees = c(50, 100, 150, 200), sample_size = c(64, 128, 256, 512))

`search` A `character` value for the hyperparameter search strategy: "random" (default), "grid".

`tune_length` A `numeric` value (integer) for either the maximum number of hyperparameter combinations ("random") or individual hyperparameter depth ("grid"): 100 (default).

Value

A list containing the following components:

`df` The data frame with anomalies removed.

`model` The best model from the grid search used to detect anomalies.

`anom_score` A data frame of predicted anomaly scores.

Examples

```
## Import data.
data(ph_crocs)

## Remove anomalies with autoencoder.
rm_outs <- ph_anomaly(df = ph_crocs, ids_col = "Biosample",
                      class_col = "Species", method = "ae")
## Alternatively, remove anomalies with extended isolation forest. Notice
## that port is defined, because running H2O sessions one after another
## can return connection errors.
rm_outs <- ph_anomaly(df = ph_crocs, ids_col = "Biosample",
                      class_col = "Species", method = "iso",
                      port = 50001)
```

`ph_crocs` *Data from: The utility of cranial ontogeny for phylogenetic inference: a case study in crocodylians using geometric morphometrics*

Description

Abstract: The degree to which the ontogeny of organisms could facilitate our understanding of phylogenetic relationships has long been a subject of contention in evolutionary biology. The famed notion that ‘ontogeny recapitulates phylogeny’ has been largely discredited, but there remains an expectation that closely related organisms undergo similar morphological transformations throughout ontogeny. To test this assumption, we used three-dimensional geometric morphometric methods to characterize the cranial morphology of 10 extant crocodylian species and construct allometric trajectories that model the post-natal ontogenetic shape changes. Using time-calibrated molecular and morphological trees, we employed a suite of comparative phylogenetic methods to assess the extent of phylogenetic signal in these trajectories. All analyses largely demonstrated a lack of significant phylogenetic signal, indicating that ontogenetic shape changes contain little phylogenetic information. Notably, some Mantel tests yielded marginally significant results when analysed with the morphological tree, which suggest that the underlying signal in these trajectories is correlated with similarities in the adult cranial morphology. However, despite these instances, all other analyses, including more powerful tests for phylogenetic signal, recovered statistical and visual evidence against the assumption that similarities in ontogenetic shape changes are commensurate with phylogenetic relatedness and thus bring into question the efficacy of using allometric trajectories for phylogenetic inference.

Usage

```
ph_crocs
```

Format

`ph_crocs`**:**

A data frame of Procrustes superimposed shape data with 183 rows and 236 columns:

Biosample Biosample

Species Species ...

Source

Downloaded from

---

ph_ctrl *Parameters for resampling and training a dataset.*

Description

The `ph_ctrl` function automatically generates a `trControl` object. This can be used in the `train` function to automatically tune hyperparameters for every classification model in the ensemble.

Usage

```
ph_ctrl(
  class,
  resample_method = "boot",
  number = ifelse(grepl("cv", resample_method, ignore.case = TRUE), 10, 25),
  repeats = ifelse(grepl("dcv$", resample_method, ignore.case = TRUE), 3, NA),
  search = "random",
  sampling = NULL
)
```

Arguments

`class` A `factor` value for training data classes.

`resample_method` A `character` value for the resampling training method: "boot" (default), "cv", LOOCV", "repeatedcv".

`number` An `integer` value for the number of resampling iterations (25 default for boot) or folds (10 default for cross-validation).

`repeats` An `integer` value for the number of sets of folds for repeated cross-validation.

`search` A `character` value for the hyperparameter search strategy: "random" (default), "grid".

`sampling` A `character` value for the sampling strategy, sometimes used to fix class imbalances: `NULL` (default), "up", "down", "smote".

Value

A `trainControl` object for the `train` function.

Examples

```
## Import data.
data(ph_crocs)
## Echo control object for train function.
ctrl <- ph_ctrl(ph_crocs$Species, resample_method = "boot")
```

ph_ensemble *Classify phenotypes via ensemble learning.*

Description

The ph_ensemble function uses classification predictions from a list of algorithms to train an ensemble model. This can be a list of manually trained algorithms from train or, more conveniently, the output from ph_train. The hyperparameter tuning and model evaluations are handled internally to simplify the ensembling process. This function assumes some preprocessing has been performed, hence the training, validation, and test set requirements.

Usage

```
ph_ensemble(
  train_models,
  train_df,
  vali_df,
  test_df,
  class_col,
  ctrl,
  train_seed = 123,
  n_cores = 2,
  task = "multi",
  metric = ifelse(task == "multi", "Kappa", "ROC"),
  top_models = 3,
  metalearner = ifelse(task == "multi", "glmnet", "rf"),
  tune_length = 10,
  quiet = FALSE
)
```

Arguments

train_models	A list of at least two train models.
train_df	A data.frame containing a class column and the training data.
vali_df	A data.frame containing a class column and the validation data.
test_df	A data.frame containing a class column and the test data.
class_col	A character value for the name of the class column. This should be consistent across data frames.
ctrl	A list containing the resampling strategy (e.g., "boot") and other parameters for trainControl. Automatically create one via ph_ctrl or manually create it with trainControl.
train_seed	A numeric value to set the training seed and control the randomness of creating resamples: 123 (default).
n_cores	An integer value for the number of cores to include in the cluster: 2 (default). We highly recommend increasing this value to, e.g., parallel::detectCores() - 1.

`task` A `character` value for the type of classification `task`: "multi" (default), "binary".

`metric` A `character` value for which summary metric should be used to select the optimal model: "ROC" (default for "binary") and "Kappa" (default for "multi"). Other options include "logLoss", "Accuracy", "Mean_Balanced_Accuracy", and "Mean_F1".

`top_models` A `numeric` value for the top n training models to ensemble: 3 (default). Every training model is ordered according to their final metric value (e.g., "ROC" or "Kappa") and the top n models are selected.

`metalearner` A `character` value for the algorithm used to train the ensemble: "glmnet" (default), "rf". Other methods, such as those listed in ph_train methods, may also be used.

`tune_length` If `search = "random"` (default), this is an `integer` value for the maximum number of hyperparameter combinations to test for each training model in the ensemble; if `search = "grid"`, this is an `integer` value for the number of levels of each hyperparameter to test for each model.

`quiet` A `logical` value for whether progress should be printed: TRUE (default), FALSE.

Value

A list containing the following components:

`ensemble_test_preds` The ensemble predictions for the test set.

`vali_preds` The validation predictions for the top models.

`test_preds` The test predictions for the top models.

`all_test_preds` The test predictions for every successfully trained model.

`all_test_results` The confusion matrix results obtained from comparing the model test predictions (i.e., original mode

`ensemble_model` The ensemble `train` object.

`var_imps` The ensemble variable importances obtained via weighted averaging. The original train importances

`train_df` The training data frame.

`vali_df` The validation data frame.

`test_df` The test data frame.

`train_models` The `train` models for the ensemble.

`ctrl` A `trainControl` object.

`metric` The summary metric used to select the optimal model.

`task` The type of classification task.

`tune_length` The maximum number of hyperparameter combinations ("random") or individual hyperparameter de

`top_models` The number of top methods selected for the ensemble.

`metalearner` The algorithm used to train the ensemble.

Examples

```
## Import data.
data(ph_crocs)

## Remove anomalies with autoencoder.
rm_outs <- ph_anomaly(df = ph_crocs, ids_col = "Biosample",
                      class_col = "Species", method = "ae")
## Preprocess anomaly-free data frame into train, validation, and test sets
## with PCs as predictors.
pc_dfs <- ph_prep(df = rm_outs$df, ids_col = "Biosample",
                  class_col = "Species", vali_pct = 0.15,
                  test_pct = 0.15, method = "pca")
## Echo control object for train function.
ctrl <- ph_ctrl(ph_crocs$Species, resample_method = "boot")
## Train all models for ensemble.
## Note: Increasing n_cores will dramatically reduce train time.
train_models <- ph_train(train_df = pc_dfs$train_df,
                         vali_df = pc_dfs$vali_df,
                         test_df = pc_dfs$test_df,
                         class_col = "Species",
                         ctrl = ctrl,
                         task = "multi",
                         methods = "all",
                         tune_length = 5,
                         quiet = FALSE)
## You can also train just a few, although more is preferable.
## Note: Increasing n_cores will dramatically reduce train time.
train_models <- ph_train(train_df = pc_dfs$train_df,
                         vali_df = pc_dfs$vali_df,
                         test_df = pc_dfs$test_df,
                         class_col = "Species",
                         ctrl = ctrl,
                         task = "multi",
                         methods = c("lda", "mda",
                         "nnet", "pda", "sparseLDA"),
                         tune_length = 5,
                         quiet = FALSE)
## Train the ensemble.
## Note: Increasing n_cores will dramatically reduce train time.
ensemble_model <- ph_ensemble(train_models = train_models$train_models,
                              train_df = pc_dfs$train_df,
                              vali_df = pc_dfs$vali_df,
                              test_df = pc_dfs$test_df,
                              class_col = "Species",
                              ctrl = ctrl,
                              task = "multi",
                              top_models = 3,
                              metalearner = "glmnet",
                              tune_length = 25,
                              quiet = FALSE)
```

ph_equate *Equate factors levels.*

Description

The ph_equate function ensures that the factor levels in all columns are equal. When classification are heavily biased or inaccurate, they can return new class predictions that do not contain every level in the original data. This can interfere with model evaluation functions e.g. via a confusion matrix.

Usage

```
ph_equate(df, class)
```

Arguments

df	A data.frame of column-wise class predictions.
class	A factor value for the observed or classes.

Value

A data frame of column-wise class predictions with class levels equal to the observed class.

Examples

```
## Make data frame of predicted classes with different levels.
## An internal or external column should contain the observed
## classes with every possible level.
obs <- as.factor(c("A", "C", "B", "D", "E"))
method_a <- c("A", "B", "B", "C", "D")
method_b <- c("A", "C", "B", "D", "C")
method_c <- c("A", "C", "B", "B", "C")
df <- data.frame(method_a, method_b, method_c)
df <- ph_equate(df = df, class = obs)
```

`ph_eval` *Evaluate a phenotype classification model.*

Description

The `ph_eval` function generates a confusion matrix for binary or multi-class classification; for the multi-class case, the results are averaged across all class levels.

Usage

```
ph_eval(pred, obs)
```

Arguments

`pred` A `factor` value of predicted classes.

`obs` A `factor` value of the observed or actual classes.

Value

A `data.frame` of confusion matrix evaluation results; for the multi-class case, the results are averaged across all class levels.

Examples

```
## Import data.
data(ph_crocs)

## Remove anomalies with autoencoder.
rm_outs <- ph_anomaly(df = ph_crocs, ids_col = "Biosample",
                      class_col = "Species", method = "ae")
## Preprocess anomaly-free data frame into train, validation, and test sets
## with PCs as predictors.
pc_dfs <- ph_prep(df = rm_outs$df, ids_col = "Biosample",
                  class_col = "Species", vali_pct = 0.15,
                  test_pct = 0.15, method = "pca")
## Echo control object for train function.
ctrl <- ph_ctrl(ph_crocs$Species, resample_method = "boot")
## Train a few models for ensemble, although more is preferable.
## Note: Increasing n_cores will dramatically reduce train time.
train_models <- ph_train(train_df = pc_dfs$train_df,
                         vali_df = pc_dfs$vali_df,
                         test_df = pc_dfs$test_df,
                         class_col = "Species",
                         ctrl = ctrl,
                         task = "multi",
                         methods = c("lda", "mda",
                         "nnet", "pda", "sparseLDA"),
                         tune_length = 5,
                         quiet = FALSE)
```

```
## Evaluate e.g. the first model.
test_pred <- predict(train_models$train_models[[1]], pc_dfs$test_df)
test_obs <- as.factor(pc_dfs$test_df$Species)
test_cm <- ph_eval(pred = test_pred, obs = test_obs)
```

`ph_iqr` *Compute interquartile range.*

Description

The `ph_iqr` function computes the interquartile range.

Usage

```
ph_iqr(x, na.rm = FALSE, type = 7)
```

Arguments

`x` A `numeric` vector.

`na.rm` A `logical` value: FALSE (default). If true, any NA is removed before quantiles are computed.

`type` An `integer` value (1:9) selecting one of 9 quantile algorithms.

Value

The interquartile range.

`ph_outs` *Find outlier indices.*

Description

The `ph_outs` function computes outliers with the interquartile method.

Usage

```
ph_outs(x)
```

Arguments

`x` A `numeric` vector.

Value

The outlier indices.

ph_prep *Preprocessing for phenotype classification via ensemble learning.*

Description

The ph_prep function splits a data frame into training, validation, and test sets, all while ensuring that every class is represented in each dataset. By default, it performs a Principal Component Analysis on the training set data and projects the validation and test data into that space. If a non-linear dimensionality reduction strategy is preferred instead, an autoencoder can be used to extract deep features. Note that the parameters max_mem_size, activation, hidden, dropout_ratio, rate, search, and tune_length are NULL unless an autoencoder, method = "ae", is used. In this case, lists or vectors can be supplied to these parameters (see parameter details) to perform a grid search for the optimal hyperparameter combination. The autoencoder with the lowest reconstruction error is selected as the best model.

Usage

```
ph_prep(
  df,
  ids_col,
  class_col,
  vali_pct = 0.15,
  test_pct = 0.15,
  scale = FALSE,
  center = NULL,
  sd = NULL,
  split_seed = 123,
  method = "pca",
  pca_pct = 0.95,
  max_mem_size = "15g",
  port = 54321,
  train_seed = 123,
  hyper_params = list(),
  search = "random",
  tune_length = 100
)
```

Arguments

df — A data.frame containing a column of unique ids, a column of classes, and an arbitrary number of numeric columns.

ids_col — A character value for the name of the ids column.

class_col — A character value for the name of the class column.

vali_pct — A numeric value for the percentage of training data to use as validation data: 0.15 (default).

test_pct A numeric value for the percentage of total data to use as test data: 0.15 (default).

scale A logical value for whether to scale the data: FALSE (default). Recommended if method = "ae" and if user intends to train other models.

center Either a logical value or numeric-alike vector of length equal to the number of columns of data to scale in df, where 'numeric-alike' means that as.numeric(.) will be applied successfully if is.numeric(.) is not true: NULL (default). If scale = TRUE, this is set to TRUE and is used to subtract the mean.

sd Either a logical value or a numeric-alike vector of length equal to the number of columns of data to scale in df: NULL (default). If scale = TRUE, this is set to TRUE and is used to divide by the standard deviation.

split_seed A numeric value to set the seed and control the randomness of splitting the data: 123 (default).

method A character value for the dimensionality reduction method: "pca" (default), "ae", "none".

pca_pct If method = "pca", a numeric value for the proportion of variance to subset the PCA with: 0.95 (default).

max_mem_size If method = "ae", a character value for the memory of an h2o session: "15g" (default).

port A numeric value for the port number of the H2O server.

train_seed A numeric value to set the control the randomness of creating resamples: 123 (default).

hyper_params A list of hyperparameters to perform a grid search. the "default" list is: list(missing_values_handling = "Skip", activation = c("Rectifier", "Tanh"), hidden = list(5, 25, 50, 100, 250, 500, nrow(df_h2o)), input_dropout_ratio = c(0, 0.1, 0.2, 0.3), rate = c(0, 0.01, 0.005, 0.001)).

search If method = "ae", a character value for the hyperparameter search strategy: "random" (default), "grid".

tune_length If method = "ae", a numeric value (integer) for either the maximum number of hyperparameter combinations ("random") or individual hyperparameter depth ("grid").

Value

A list containing the following components:

train_df The training set data frame.

vali_df The validation set data frame.

test_df The test set data frame.

train_split The training set indices from the original data frame.

vali_split The validation set indices from the original data frame.

test_split	The test set indices from the original data frame.
vali_pct	The percentage of training data used as validation data.
test_pct	The percentage of total data used as test data.
method	The dimensionality reduction method.

Examples

```
## Import data.
data(ph_crocs)

## Remove anomalies with autoencoder.
rm_outs <- ph_anomaly(df = ph_crocs, ids_col = "Biosample",
                      class_col = "Species", method = "ae")
## Preprocess anomaly-free data frame into train, validation, and test sets
## with PCs as predictors.
pc_dfs <- ph_prep(df = rm_outs$df, ids_col = "Biosample",
                  class_col = "Species", vali_pct = 0.15,
                  test_pct = 0.15, method = "pca")
## Alternatively, preprocess data frame into train, validation, and test
## sets with latent variables as predictors. Notice that port is defined,
## because running H2O sessions one after another can cause connection
## errors.
ae_dfs <- ph_prep(df = rm_outs$df, ids_col = "Biosample", class_col = "Species",
                  vali_pct = 0.15, test_pct = 0.15, method = "ae", port = 50001)
```

ph_stickleback *Data from: Sexually mediated phenotypic variation within and between sexes as a continuum structured by ecology: The mosaic nature of skeletal variation across body regions in Threespine stickleback (Gasterosteus aculeatus L.)*

Description

Abstract: Ecological character displacement between the sexes, and sexual selection, integrate into a convergent set of factors that produce sexual variation. Ecologically-modulated, sexually mediated variation within and between sexes may be a major contributor to the amount of total variation that selection can act on in species. Threespine stickleback (Gasterosteus aculeatus) display rapid adaptive responses and sexual variation in many phenotypic traits. We examined phenotypic variation in the skull, pectoral and pelvic girdles of threespine stickleback from two freshwater and two coastal marine sites on the Sunshine Coast of British Columbia, Canada, using an approach that avoids a priori assumptions about bimodal patterns of variation. We quantified shape and size of the cranial, pectoral and pelvic regions of sticklebacks in marine and freshwater habitats using 3D geometric morphometrics and an index of sexually mediated variation. We show that the expression of phenotypic variation is structured in part by the effects of both habitat marine vs freshwater and the effects of individual sites within each habitat. Relative size exerts variable influence, and patterns of phenotypic variation associated with sex vary among body regions. This fine-grained quantification of sexually mediated variation in the context of habitat difference and different anatomical structures indicates a complex relationship between genetically inferred sex and environmental factors, demonstrating that the interplay between shared genetic background and sexually mediated, ecologically- based selective pressures structures the phenotypic expression of complex traits.

Usage

```
ph_stickleback
```

Format

`ph_stickleback`**:**

A data frame of Procrustes superimposed shape data with 190 rows and 214 columns:

Biosample Biosample

Habitat Habitat

Population Population

Sex Sex ...

Source

Downloaded from <doi:doi.org/10.5061/dryad.xd2547dkw>

`ph_train` *Generate predictions for phenotype ensemble.*

Description

The `ph_train` function automatically trains a set of binary or multi-class classification models to ultimately build a new dataset of predictions. The data preprocessing and hyperparameter tuning are handled internally to minimize user input and simplify the training.

Usage

```
ph_train(
  train_df,
  vali_df,
  test_df,
  class_col,
  ctrl,
  train_seed = 123,
  n_cores = 2,
  task = "multi",
  methods = "all",
  metric = ifelse(task == "multi", "Kappa", "ROC"),
  tune_length = 10,
  quiet = FALSE
)
```

Arguments

train_df A `data.frame` containing a class column and the training data.

vali_df A `data.frame` containing a class column and the validation data.

test_df A `data.frame` containing a class column and the test d

class_col A `character` value for the name of the class column shared across the train, validation, and test sets.

ctrl A `list` containing the resampling strategy (e.g., "boot") and other parameters for `trainControl`. Automatically create one via `ph_ctrl` or manually create it with `trainControl`.

train_seed A `numeric` value to set the training seed and control the randomness of creating resamples: 123 (default).

n_cores An `integer` value for the number of cores to include in the cluster: 2 (default). We highly recommend increasing this value to, e.g., parallel::detectCores() - 1.

task A `character` value for the type of classification `task`: "multi" (default), "binary".

methods A `character` value enumerating the names (at least two, unless "all") of the classification methods to ensemble: "all" (default).

- If `task = "binary"`, there are 33 methods to choose from: "AdaBag", "AdaBoost.M1", "C5.0", "evtree", "glmnet", "hda", "kernelpls", "kknn", "lda", "loclda", "mda", "nb", "nnet", "pda", "pls", "qda", "rda", "rf", "sparseLDA", "stepLDA", "stepQDA", "treebag", "svmLinear", "svmPoly","svmRadial", "gaussprLinear" (slow), "gaussprPoly" (slow), "gaussprRadial" (slow), "bagEarthGCV", "cforest", "earth", "fda", "hdda".
- If `task = "multi"`, there are 30 methods to choose from: "AdaBag", "AdaBoost.M1", "C5.0", "evtree", "glmnet", "hda", "kernelpls", "kknn", "lda", "loclda", "mda", "nb", "nnet", "pda", "pls", "qda", "rda", "rf", "sparseLDA", "stepLDA", "stepQDA", "treebag", "svmLinear", "svmPoly", "svmRadial", "bagEarthGCV", "cforest", "earth", "fda", "hdda".

metric A `character` value for which summary metric should be used to select the optimal model: "ROC" (default for "binary") and "Kappa" (default for "multi"). Other options include "logLoss", "Accuracy", "Mean_Balanced_Accuracy", and "Mean_F1".

tune_length If `search = "random"` (default), this is an `integer` value for the maximum number of hyperparameter combinations to test for each training model in the ensemble; if `search = "grid"`, this is an `integer` value for the number of levels of each hyperparameter to test for each model.

quiet A `logical` value for whether progress should be printed: TRUE (default), FALSE.

Value

A list containing the following components:

`train_models` The `train` models for the ensemble.

`train_df` The training data frame.

`vali_df` The validation data frame.

`test_df` The test data frame.

`task` The type of classification task.

`ctrl` A list of resampling parameters used in `trainControl`.

`methods` The names of the classification methods to ensemble.

`search` The hyperparameter search strategy.

`n_cores` The number of cores for parallel processing.

`metric` The summary metric used to select the optimal model.

`tune_length` The maximum number of hyperparameter combinations ("random") or individual hyperparameter depth ("gri

Examples

```
## Import data.
data(ph_crocs)

## Remove anomalies with autoencoder.
rm_outs <- ph_anomaly(df = ph_crocs, ids_col = "Biosample",
                      class_col = "Species", method = "ae")
## Preprocess anomaly-free data frame into train, validation, and test sets
## with PCs as predictors.
pc_dfs <- ph_prep(df = rm_outs$df, ids_col = "Biosample",
                  class_col = "Species", vali_pct = 0.15,
                  test_pct = 0.15, method = "pca")
## Echo control object for train function.
ctrl <- ph_ctrl(ph_crocs$Species, resample_method = "boot")
## Train all models for ensemble.
## Note: Increasing n_cores will dramatically reduce train time.
train_models <- ph_train(train_df = pc_dfs$train_df,
                         vali_df = pc_dfs$vali_df,
                         test_df = pc_dfs$test_df,
                         class_col = "Species",
                         ctrl = ctrl,
                         task = "multi",
                         methods = "all",
                         tune_length = 5,
                         quiet = FALSE)
## You can also train just a few, although more is preferable.
## Note: Increasing n_cores will dramatically reduce train time.
train_models <- ph_train(train_df = pc_dfs$train_df,
                         vali_df = pc_dfs$vali_df,
                         test_df = pc_dfs$test_df,
                         class_col = "Species",
                         ctrl = ctrl,
                         task = "multi",
                         methods = c("lda", "mda",
                         "nnet", "pda", "sparseLDA"),
                         tune_length = 5,
                         quiet = FALSE)
```

Index