Package ‘MantaID’

January 20, 2025

Title A Machine-Learning Based Tool to Automate the Identification of Biological Database IDs

Version 1.0.4

Description The number of biological databases is growing rapidly, but different databases use different IDs to refer to the same biological entity. The inconsistency in IDs impedes the integration of various types of biological data. To resolve the problem, we developed 'MantaID', a data-driven, machine-learning based approach that automates identifying IDs on a large scale. The 'MantaID' model's prediction accuracy was proven to be 99%, and it correctly and effectively predicted 100,000 ID entries within two minutes. 'MantaID' supports the discovery and exploitation of ID patterns from large quantities of databases. (e.g., up to 542 biological databases). An easy-to-use freely available open-source software R package, a user-friendly web application, and API were also developed for 'MantaID' to improve applicability. To our knowledge, 'MantaID' is the first tool that enables an automatic, quick, accurate, and comprehensive identification of large quantities of IDs, and can therefore be used as a starting point to facilitate the complex assimilation and aggregation of biological data across diverse databases.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.4.0),biomaRt,caret,keras,mlr3tuning,mlr3

LazyData true

URL https://molaison.github.io/MantaID/

Imports
ggplot2,data.table,magrittr,stringr,tibble,tidyr,tidyselect,ggcorrplot,reshape2,scutr,paradox,RColorBrewer,purrr,dplyr

Suggests mlr3hyperband,mlr3learners,ranger,rpart,xgboost

NeedsCompilation no

Author Zhengpeng Zeng [aut, cre, ctb]
(<https://orcid.org/0000-0001-7919-209X>),
Longfei Mao [aut, cph] (<https://orcid.org/0000-0003-0759-0501>),
Feng Yu [aut] (<https://orcid.org/0000-0002-5221-281X>),
Jiamin Hu [ctb] (<https://orcid.org/0000-0003-3030-2117>),
Xiting Wang [ctb] (<https://orcid.org/0009-0009-5235-0006>)

Maintainer Zhengpeng Zeng <molaison@foxmail.com>

Repository CRAN

Date/Publication 2024-09-09 07:50:02 UTC

Contents

Example 2
mi 3
mi_balance_data 4
mi_clean_data 4
mi_data_attributes 5
mi_data_procID 5
mi_data_rawID 6
mi_filter_feat 6
mi_get_confusion 7
mi_get_ID 7
mi_get_ID_attr 8
mi_get_importance 8
mi_get_miss 9
mi_get_padlen 9
mi_plot_cor 10
mi_plot_heatmap 10
mi_predict_new 11
mi_run_bmr 11
mi_split_col 12
mi_split_str 12
mi_to_numer 13
mi_train_BP 13
mi_train_rg 14
mi_train_rp 15
mi_train_xgb 15
mi_tune_rg 16
mi_tune_rp 16
mi_tune_xgb 17
mi_unify_mod 18

Index **19**

`Example` *ID example dataset.*

Description

ID example dataset.

Usage

```
Example
```

Format

A tibble with 5000 rows and 2 variables.

ID A identifier character.

class The database the ID belongs to.

`mi` *A wrapper function that executes MantaID workflow.*

Description

A wrapper function that executes MantaID workflow.

Usage

```
mi(
  cores = NULL,
  levels = c("*", 0:9, letters, LETTERS, "_", ".", "-", " ", "/", "\\", ":"),
  ratio = 0.3,
  para_blc = FALSE,
  model_path = NULL,
  batch_size = 128,
  epochs = 64,
  validation_split = 0.3,
  graph_path = NULL
)
```

Arguments

`cores` The number of cores used when balancing data.

`levels` The vector that includes all the single characters occurred in IDs.

`ratio` The ratio of the test set.

`para_blc` A logical value whether using parallel computing when balancing data.

`model_path` The path to save models.

`batch_size` The batch size of deep learning model fitting.

`epochs` The epochs of deep learning model fitting.

`validation_split` The validation ratio of deep learning model fitting.

`graph_path` The path to save graphs.

Value

The list of models and graphs.

mi_balance_data *Data balance. Most classes adopt random undersampling, while a few classes adopt smote method to oversample to obtain relatively balanced data.*

Description

Data balance. Most classes adopt random undersampling, while a few classes adopt smote method to oversample to obtain relatively balanced data.

Usage

```
mi_balance_data(data, ratio = 0.3, parallel = FALSE)
```

Arguments

`data`	A data frame. Except for class column, all are numeric types.
`ratio`	Numeric between 0 and 1. The percent of the test set split from data.
`parallel`	Logical.

Value

A list contains a train set and a test set.

mi_clean_data *Reshape data and delete meaningless rows.*

Description

Reshape data and delete meaningless rows.

Usage

```
mi_clean_data(data, cols = everything(), placeholder = c("-"))
```

Arguments

`data`	A dataframe or tibble or data.table or matrix. Names of the column will be regard as the class of ID included in column.
`cols`	Character vectors. Columns of `data` that contain the IDs
`placeholder`	Character vectors. IDs included in `placeholder` will be omitted.

Value

A tibble with two columns("ID" and "class")

Examples

```
data <- tibble::tibble(
  "class1" = c("A", "B", "C", "D"),
  "class2" = c("E", "F", "G", "H"),
  "class3" = c("L", "M", "-", "O")
)
mi_clean_data(data)
```

`mi_data_attributes` *ID-related datasets in biomart.*

Description

ID-related datasets in biomart.

Usage

```
mi_data_attributes
```

Format

A dataframe with 65 variables and 3 variables.

name The name of dataset.

description Description of dataset.

page collection of attributes.

`mi_data_procID` *Processed ID data.*

Description

Processed ID data.

Usage

```
mi_data_procID
```

Format

A tibble dataframe with 5000 rows and 21 variables.

pos1 to pos20 Splited ID.

class The databases that ID belongs to.

mi_data_rawID *ID dataset for testing.*

Description

ID dataset for testing.

Usage

```
mi_data_rawID
```

Format

A tibble with 5000 rows and 2 variables.

ID A identifier character.

class The database the ID belongs to.

mi_filter_feat *Performing feature selection in a automatic way based on correlation and feature importance.*

Description

Performing feature selection in a automatic way based on correlation and feature importance.

Usage

```
mi_filter_feat(data, cor_thresh = 0.7, imp_thresh = 0.99, union = FALSE)
```

Arguments

`data`	The data frame returned by `mi_to_numer()`.
`cor_thresh`	The threshold set for Pearson correlation. If correlation value is over this threshold, the two features will be viewed as redundant and one of them will be removed.
`imp_thresh`	The threshold set for feature importance. The last several features with the lowest importance will be removed if remained importance lower than `imp_thresh`.
`union`	The method for combining the decisions of correlation method and importance method. If `TRUE`, any of the features calculated by the two methods will be returned. Otherwise, only features in the results of both methods will be returned.

Value

The names of the features that should be removed.

`mi_get_confusion` *Compute the confusion matrix for the predicted result.*

Description

Compute the confusion matrix for the predicted result.

Usage

```
mi_get_confusion(result_list, ifnet = FALSE)
```

Arguments

`result_list`	A list returned from model training functions.
`ifnet`	Logical.Whether the data is obtained by a deep learning model.

Value

A `confusionMatrix` object.

`mi_get_ID` *Get ID data from the* `Biomart` *database using* `attributes`.

Description

Get ID data from the `Biomart` database using `attributes`.

Usage

```
mi_get_ID(
  attributes,
  biomart = "genes",
  dataset = "hsapiens_gene_ensembl",
  mirror = "asia"
)
```

Arguments

`attributes`	A dataframe.The information we want to retrieve.Use `mi_get_ID_attr` to hava try.
`biomart`	BioMart database name you want to connect to. Use `biomaRt::listEnsembl` to retrieve the possible database names.
`dataset`	Datasets of the selected BioMart database.
`mirror`	Specify an Ensembl mirror to connect to.

Value

A tibble dataframe.

---

mi_get_ID_attr *Get ID attributes from the* Biomart *database.*

Description

Get ID attributes from the Biomart database.

Usage

```
mi_get_ID_attr(
  biomart = "genes",
  dataset = "hsapiens_gene_ensembl",
  mirror = "asia"
)
```

Arguments

biomart	BioMart database name you want to connect to.Use biomaRt::listEnsembl to retrieve the possible database names.
dataset	Datasets of the selected BioMart database.
mirror	Specify an Ensembl mirror to connect to.

Value

A dataframe.

---

mi_get_importance *Plot the bar plot for feature importance.*

Description

Plot the bar plot for feature importance.

Usage

```
mi_get_importance(data)
```

Arguments

data	A table.

Value

A bar plot.

mi_get_miss *Observe the distribution of the false response of the test set.*

Description

Observe the distribution of the false response of the test set.

Usage

```
mi_get_miss(predict)
```

Arguments

predict	An R6 class `PredictionClassif`.

Value

A tibble data frame that records the number of wrong predictions for each category ID

mi_get_padlen *Get max length of ID data.*

Description

Get max length of ID data.

Usage

```
mi_get_padlen(data)
```

Arguments

data	A dataframe.

Value

An int object.

Examples

```
data(mi_data_rawID)
mi_get_padlen(mi_data_rawID)
```

mi_plot_cor *Plot correlation heatmap.*

Description

Plot correlation heatmap.

Usage

```
mi_plot_cor(data, cls = "class")
```

Arguments

data	Data frame including IDs' position features.
cls	The name of the class column.

Value

A heatmap.

Examples

```
data(mi_data_procID)
data_num <- mi_to_numer(mi_data_procID)
mi_plot_cor(data_num)
```

mi_plot_heatmap *Plot heatmap for result confusion matrix.*

Description

Plot heatmap for result confusion matrix.

Usage

```
mi_plot_heatmap(table, name = NULL, filepath = NULL)
```

Arguments

table	A table.
name	Model names.
filepath	File path the plot to save. Default NULL.

Value

A `ggplot` object.

mi_predict_new *Predict new data with a trained learner.*

Description

Predict new data with a trained learner.

Usage

```
mi_predict_new(data, result, ifnet = F)
```

Arguments

data	A dataframe.
result	The result object from a previous training.
ifnet	A boolean indicating if a neural network is used for prediction.

Value

A data frame that contains features and 'predict' class.

mi_run_bmr *Compare classification models with small samples.*

Description

Compare classification models with small samples.

Usage

```
mi_run_bmr(data, row_num = 1000, resamplings = rsmps("cv", folds = 10))
```

Arguments

data	A tibble.All are numeric except the first column is a factor.
row_num	The number of samples used.
resamplings	R6/Resampling.Resampling method.

Value

A list of R6 class of benchmark results and scores of test set. examples data(mi_data_procID) mi_run_bmr(mi_data_procID)

mi_split_col *Cut the string of ID column character by character and divide it into multiple columns.*

Description

Cut the string of ID column character by character and divide it into multiple columns.

Usage

```
mi_split_col(data, cores = NULL, pad_len = 10)
```

Arguments

`data`	Dataframe(tibble) to be split.
`cores`	Int.The num of cores to allocate for computing.
`pad_len`	The length of longest id, i.e. the maxlength.

Value

A tibble with pad_len+1 column.

mi_split_str *Split the string into individual characters and complete the character vector to the maximum length.*

Description

Split the string into individual characters and complete the character vector to the maximum length.

Usage

```
mi_split_str(str, pad_len)
```

Arguments

`str`	The string to be splited.
`pad_len`	The length of longest ID, i.e. the maxlength.

Value

Splited character vector.

Examples

```
string_test <- "Good Job"
length <- 15
mi_split_str(string_test, length)
```

mi_to_numer *Convert data to numeric, and for the ID column convert with fixed levels.*

Description

Convert data to numeric, and for the ID column convert with fixed levels.

Usage

```
mi_to_numer(
  data,
  levels = c("*", 0:9, letters, LETTERS, "_", ".", "-", " ", "/", "\\", ":")
)
```

Arguments

`data`	A tibble with n position column(pos1,pos2,...) and class column.
`levels`	Characters accommodated in IDs.

Value

A numeric data frame with numerical or factor type columns.

Examples

```
data(mi_data_procID)
mi_to_numer(mi_data_procID)
```

mi_train_BP *Train a three layers neural network model.*

Description

Train a three layers neural network model.

Usage

```
mi_train_BP(
  train,
  test,
  cls = "class",
  path2save = NULL,
  batch_size = 128,
  epochs = 64,
  validation_split = 0.3,
  verbose = 0
)
```

Arguments

`train` A dataframe with the `class` column as label.

`test` A dataframe with the `class` column as label.

`cls` A character.The name of the label column.

`path2save` The folder path to store the model and train history.

`batch_size` Integer or NULL. The number of samples per gradient update.

`epochs` The number of epochs to train the model.

`validation_split` Float between 0 and 1. Fraction of the training data to be used as validation data.

`verbose` The verbosity mode.

Value

A `list` object containing the prediction confusion matrix, the `model` object, and the mapping of predicted numbers to classes.

`mi_train_rg` *Random Forest Model Training.*

Description

Random Forest Model Training.

Usage

```
mi_train_rg(train, test, measure = msr("classif.acc"), instance = NULL)
```

Arguments

`train` A dataframe.

`test` A dataframe.

`measure` Model evaluation method.

`instance` A tuner.

Value

A list of learner for predicting and predicted result of test set.

mi_train_rp *Classification tree model training.*

Description

Classification tree model training.

Usage

```
mi_train_rp(train, test, measure = msr("classif.acc"), instance = NULL)
```

Arguments

train	A dataframe.
test	A dataframe.
measure	Model evaluation method.Use mlr_measures and msr() to view and choose metrics.
instance	A tuner.

Value

A list of learner for predicting and predicted result of test set.

mi_train_xgb *Xgboost model training*

Description

Xgboost model training

Usage

```
mi_train_xgb(train, test, measure = msr("classif.acc"), instance = NULL)
```

Arguments

train	A dataframe.
test	A dataframe.
measure	Model evaluation method.
instance	A tuner.

Value

A list of learner for predicting and predicted result of test set.

mi_tune_rg *Tune the Random Forest model by hyperband.*

Description

Tune the Random Forest model by hyperband.

Usage

```
mi_tune_rg(
  data,
  resampling = rsmp("cv", folds = 5),
  measure = msr("classif.acc"),
  eta = 3
)
```

Arguments

`data`	A tibble.All are numeric except the first column is a factor.
`resampling`	R6/Resampling.
`measure`	Model evaluation method.Use `mlr_measures` and `msr()` to view and choose metrics.
`eta`	The percent parameter configurations discarded.

Value

A list of tuning instance and stage plot.

mi_tune_rp *Tune the Decision Tree model by hyperband.*

Description

Tune the Decision Tree model by hyperband.

Usage

```
mi_tune_rp(
  data,
  resampling = rsmp("bootstrap", ratio = 0.8, repeats = 5),
  measure = msr("classif.acc"),
  eta = 3
)
```

Arguments

`data`	A tibble.All are numeric except the first column is a factor.
`resampling`	R6/Resampling.
`measure`	Model evaluation method.Use `mlr_measures` and `msr()` to view and choose metrics.
`eta`	The percent parameter configurations discarded.

Value

A list of tuning instance and stage plot.

`mi_tune_xgb` *Tune the Xgboost model by hyperband.*

Description

Tune the Xgboost model by hyperband.

Usage

```
mi_tune_xgb(
  data,
  resampling = rsmp("cv", folds = 5),
  measure = msr("classif.acc"),
  eta = 3
)
```

Arguments

`data`	A tibble.All are numeric except the first column is a factor.
`resampling`	R6/Resampling.
`measure`	Model evaluation method.Use `mlr_measures` and `msr()` to view and choose metrics.
`eta`	The percent parameter configurations discarded.

Value

A list of tuning instance and stage plot.

mi_unify_mod *Predict with four models and unify results by the sub-model's specificity score to the four possible classes.*

Description

Predict with four models and unify results by the sub-model's specificity score to the four possible classes.

Usage

```
mi_unify_mod(
  data,
  col_id,
  result_rg,
  result_rp,
  result_xgb,
  result_BP,
  c_value = 0.75,
  pad_len = 30
)
```

Arguments

data	A dataframe contains the ID column.
col_id	The name of ID column.
result_rg	The result from the Random Forest model.
result_rp	The result from the Decision Tree model.
result_xgb	The result from the XGBoost model.
result_BP	The result from the Backpropagation Neural Network model.
c_value	A numeric value used in the final prediction calculation.
pad_len	The length to pad the ID characters to.

Value

A dataframe.

Index

∗ **datasets**

Example, 2
mi_data_attributes, 5
mi_data_procID, 5
mi_data_rawID, 6

Example, 2

mi, 3
mi_balance_data, 4
mi_clean_data, 4
mi_data_attributes, 5
mi_data_procID, 5
mi_data_rawID, 6
mi_filter_feat, 6
mi_get_confusion, 7
mi_get_ID, 7
mi_get_ID_attr, 8
mi_get_importance, 8
mi_get_miss, 9
mi_get_padlen, 9
mi_plot_cor, 10
mi_plot_heatmap, 10
mi_predict_new, 11
mi_run_bmr, 11
mi_split_col, 12
mi_split_str, 12
mi_to_numer, 13
mi_train_BP, 13
mi_train_rg, 14
mi_train_rp, 15
mi_train_xgb, 15
mi_tune_rg, 16
mi_tune_rp, 16
mi_tune_xgb, 17
mi_unify_mod, 18