Package 'CBDA'

April 16, 2018

Type Package

Title Compressive Big Data Analytics

Version 1.0.0

Maintainer Simeone Marino <simeonem@umich.edu>

Description Classification performed on Big Data. It uses concepts from compressive sensing, and implements ensemble predictor (i.e., 'SuperLearner') and knockoff filtering as the main machine learning and feature mining engines.

License GPL-3

URL https://github.com/SOCR/CBDA

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports stats , utils , prettydoc , foreach , SuperLearner, parallel , doParallel

Depends R(>= 3.3.0)

VignetteBuilder knitr

Suggests knitr, rmarkdown , FNN , e1071 , missForest , knockoff , caret , smotefamily , xgboost , bartMachine , glmnet , randomForest

NeedsCompilation no

Author Simeone Marino [aut, cre], Ivo Dinov [aut]

Repository CRAN

Date/Publication 2018-04-16 14:56:33 UTC

R topics documented:

CBDA *Main Compressive Big Data Analytics - CBDA function*

Description

This CBDA function comprises all the input specifications to run a set M of subsamples from the Big Data [Xtemp, Ytemp]. We assume that the Big Data is already clean and harmonized. This version 1.0.0 is fully tested ONLY on continuous features Xtemp and binary outcome Ytemp.

Usage

```
CBDA(Ytemp, Xtemp, label = "CBDA_package_test", alpha = 0.2, Kcol_min = 5,
  Kcol_max = 15, Nrow_min = 30, Nrow_max = 50, misValperc = 0,
  M = 3000, N_cores = 1, top = 1000, workspace_directory = tempdir(),
  max_covs = 100, min_covs = 5, algorithm_list = c("SL.glm", "SL.xgboost",
  "SL.glmnet", "SL.svm", "SL.randomForest", "SL.bartMachine"))
```

Arguments

`Ytemp`	This is the output variable (vector) in the original Big Data
`Xtemp`	This is the input variable (matrix) in the original Big Data
`label`	This is the label appended to RData workspaces generated within the CBDA calls
`alpha`	Percentage of the Big Data to hold off for Validation
`Kcol_min`	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
`Kcol_max`	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
`Nrow_min`	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
`Nrow_max`	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)

`misValperc` Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).

`M` Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining

`N_cores` Number of Cores to use in the parallel implementation (default is set to 1 core)

`top` Top predictions to select out of the M (must be < M, optimal ~0.1*M)

`workspace_directory` Directory where the results and workspaces are saved (set by default to tempdir())

`max_covs` Top features to display and include in the Validation Step where nested models are tested

`min_covs` Minimum number of top features to include in the initial model for the Validation Step (it must be greater than 2)

`algorithm_list` List of algorithms/wrappers used by the SuperLearner. By default is set to the following list algorithm_list <- c("SL.glm","SL.xgboost", "SL.glmnet","SL.svm","SL.randomForest","SL

Details

This function comprises all the input specifications to run a set M of subsamples from the Big Data [Xtemp, Ytemp]. We assume that the Big Data is already clean and harmonized. After the necessary data wrangling (i.e., imputation, normalization and rebalancing), an ensemble predictor (i.e., SuperLearner) is applied to each subsample for training/learning. The list of algorithms used by the SuperLearner is supplied by an external file to be placed in the working directory (e.g.: CBDA_SL_library.m in our release). The file can contain any SuperLearner wrapper and any wrappers properly defined by the user. The ensemble predictive model is then validated on a fraction alpha of the Big Data. Each subsample generates a predictive model that is ranked based on performance metrics (e.g., Mean Square Error-MSE and Accuracy) during the first validation step. After all the M subsamples have been generated and each predictive model computed, the CBDA function calls 4 more functions to perform i) CONSOLIDATION and ranking of the results where the top predictive models are selected (top) and the more frequent features (BEST) are ranked and displayed as well, ii) VALIDATION on the top ranked features (i.e., up to "max_covs" number of features) where nested ensemble predictive models are generated in a bottom-up fashion, iii) Implementation of STOPPING CRITERIA for the best/optimal ensemble predictive model (to avoid overfitting) and iv) CLEAN UP step for deleting unnecessary workspaces generated by the CBDA protocol. IMPORTANT - Memory limits to run CBDA: see https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html for various limitations on memory needs while running R under different OS. As far as CBDA is concerned, a CBDA object can be up to 200-300 Mb. The space needed to save all the workspaces however may need to be as large as 1-5 Gb, depending on the number of subsamples. We are working on an new CBDA implementation that reduces the storage constraints.

Value

CBDA object with validation results and 3 RData workspaces

References

See https://github.com/SOCR/CBDA/releases for details on the CBDA protocol and the manuscript "Controlled Feature Selection and Compressive Big Data Analytics: Applications to Big Biomedical and Health Studies" [under review] authored by Simeone Marino, Jiachen Xu, Yi Zhao, Nina Zhou, Yiwang Zhou, Ivo D. Dinov from the University of Michigan

Examples

```
# Installation
# Please upload the Windows binary and/or source CBDA_1.0.0 files from
# the CBDA Github repository https://github.com/SOCR/CBDA/releases
## Not run:
# Installation from the Windows binary (recommended for Windows systems)
install.packages("/filepath/CBDA_1.0.0_binary_Windows.zip", repos = NULL, type = "win.binary")

# Installation from the source (recommended for Macs and Linux systems)
install.packages("/filepath/CBDA_1.0.0_source_.tar.gz", repos = NULL, type = "source")

# Initialization
# This function call installs (if needed) and attaches all the necessary packages to run
# the CBDA package v1.0.0. It should be run before any production run or test.
# The output shows a table where for each package a TRUE or FALSE is displayed.
# Thus the necessary steps can be pursued in case some package has a FALSE.
CBDA_initialization()

# Set the specs for the synthetic dataset to be tested
n = 300          # number of observations
p = 100          # number of variables

# Generate a nxp matrix of IID variables (e.g., ~N(0,1))
X1 = matrix(rnorm(n*p), nrow=n, ncol=p)

# Setting the nonzero variables - signal variables
nonzero=c(1,100,200,300,400,500,600,700,800,900)

# Set the signal amplitude (for noise level = 1)
amplitude = 10

# Allocate the nonzero coefficients in the correct places
beta = amplitude * (1:p %in% nonzero)

# Generate a linear model with a bias (e.g., white  noise ~N(0,1))
ztemp <- function() X1 %*% beta + rnorm(n)
z = ztemp()

# Pass it through an inv-logit function to
# generate the Bernoulli response variable Ytemp
pr = 1/(1+exp(-z))
Ytemp = rbinom(n,1,pr)
X2 <- cbind(Ytemp,X1)

dataset_file ="Binomial_dataset_3.txt"
```

```
# Save the synthetic dataset
a <- tempdir()
write.table(X2, file = paste0(file.path(a),'/',dataset_file), sep=",")

# The file is now stored in the directory a
a
list.files(a)

# Load the Synthetic dataset
Data = read.csv(paste0(file.path(a),'/',dataset_file),header = TRUE)
Ytemp <- Data[,1] # set the outcome
original_names_Data <- names(Data)
cols_to_eliminate=1
Xtemp <- Data[-cols_to_eliminate] # set the matrix X of features/covariates
original_names_Xtemp <- names(Xtemp)

# Add more wrappers/algorithms to the SuperLearner ensemble predictor
# It can be commented out if only the default set of algorithms are used,
# e.g., algorithm_list = c("SL.glm","SL.xgboost","SL.glmnet","SL.svm",
#                          "SL.randomForest","SL.bartMachine")
# This defines a "new" wrapper, based on the default SL.glmnet
 SL.glmnet.0.75 <- function(..., alpha = 0.75,family="binomial"){
                 SL.glmnet(..., alpha = alpha, family = family)}

 test_example <- c("SL.glmnet","SL.glmnet.0.75")

# Call the Main CBDA function
# Multicore functionality NOT enabled
CBDA_object <- CBDA(Ytemp , Xtemp , M = 12 , Nrow_min = 50, Nrow_max = 70,
              top = 10, max_covs = 8 , min_covs = 3,algorithm_list = test_example ,
              workspace_directory = a)

# Multicore functionality enabled
test_example <- c("SL.xgboost","SL.svm")
CBDA_test <- CBDA(Ytemp , Xtemp , M = 40 , Nrow_min = 50, Nrow_max = 70,
               N_cores = 2 , top = 30, max_covs = 20 ,
                min_covs = 5 , algorithm_list = test_example ,
              workspace_directory = a)

## End(Not run)
```

CBDA.pipeline *Training/Leaning Step for Compressive Big Data Analytics - LONI PIPELINE*

Description

The CBDA.pipeline() function comprises all the input specifications to run a set M of subsamples from the Big Data [Xtemp, Ytemp]. We assume that the Big Data is already clean and harmonized.

This version 1.0.0 is fully tested ONLY on continuous features Xtemp and binary outcome Ytemp.

Usage

```
CBDA.pipeline(job_id, Ytemp, Xtemp, label = "CBDA_package_test",
  alpha = 0.2, Kcol_min = 5, Kcol_max = 15, Nrow_min = 30,
  Nrow_max = 50, misValperc = 0, M = 3000, N_cores = 1, top = 1000,
  workspace_directory = setwd(tempdir()), max_covs = 100, min_covs = 5,
  algorithm_list = c("SL.glm", "SL.xgboost", "SL.glmnet", "SL.svm",
  "SL.randomForest", "SL.bartMachine"))
```

Arguments

job_id	This is the ID for the job generator in the LONI pipeline interface
Ytemp	This is the output variable (vector) in the original Big Data
Xtemp	This is the input variable (matrix) in the original Big Data
label	This is the label appended to RData workspaces generated within the CBDA calls
alpha	Percentage of the Big Data to hold off for Validation
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
N_cores	Number of Cores to use in the parallel implementation (default is set to 1 core)
top	Top predictions to select out of the M (must be < M, optimal ~0.1*M)
workspace_directory	Directory where the results and workspaces are saved (set by default to tempdir())
max_covs	Top features to display and include in the Validation Step where nested models are tested
min_covs	Minimum number of top features to include in the initial model for the Validation Step (it must be greater than 2)
algorithm_list	List of algorithms/wrappers used by the SuperLearner. By default is set to the following list algorithm_list <- c("SL.glm","SL.xgboost", "SL.glmnet","SL.svm","SL.randomForest","SL

Value

CBDA object with validation results and 3 RData workspaces

CBDA.training *Training/Learning Compressive Big Data Analytics - CBDA.training function*

Description

This CBDA function comprises all the input specifications to run a set M of subsamples from the Big Data [Xtemp, Ytemp]. We assume that the Big Data is already clean and harmonized. This version 1.0.0 is fully tested ONLY on continuous features Xtemp and binary outcome Ytemp. It only performs the Training/Learning step of the CBDA protocol.

Usage

```
CBDA.training(Ytemp, Xtemp, label = "CBDA_package_test", alpha = 0.2,
  Kcol_min = 5, Kcol_max = 15, Nrow_min = 30, Nrow_max = 50,
  misValperc = 0, M = 3000, N_cores = 1, top = 1000,
  workspace_directory = tempdir(), max_covs = 100, min_covs = 5,
  algorithm_list = c("SL.glm", "SL.xgboost", "SL.glmnet", "SL.svm",
  "SL.randomForest", "SL.bartMachine"))
```

Arguments

Ytemp	This is the output variable (vector) in the original Big Data
Xtemp	This is the input variable (matrix) in the original Big Data
label	This is the label appended to RData workspaces generated within the CBDA calls
alpha	Percentage of the Big Data to hold off for Validation
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
N_cores	Number of Cores to use in the parallel implementation (default is set to 1 core)
top	Top predictions to select out of the M (must be < M, optimal ~0.1*M)
workspace_directory	Directory where the results and workspaces are saved (set by default to tempdir())

max_covs	Top features to display and include in the Validation Step where nested models are tested
min_covs	Minimum number of top features to include in the initial model for the Validation Step (it must be greater than 2)
algorithm_list	List of algorithms/wrappers used by the SuperLearner. By default is set to the following list algorithm_list <- c("SL.glm","SL.xgboost", "SL.glmnet","SL.svm","SL.randomForest","SL

Details

This function comprises all the input specifications to run a set M of subsamples from the Big Data [Xtemp, Ytemp]. We assume that the Big Data is already clean and harmonized. After the necessary data wrangling (i.e., imputation, normalization and rebalancing), an ensemble predictor (i.e., SuperLearner) is applied to each subsample for training/learning. The list of algorithms used by the SuperLearner is supplied by an external file to be placed in the working directory (e.g.: CBDA_SL_library.m in our release). The file can contain any SuperLearner wrapper and any wrappers properly defined by the user. The ensemble predictive model is then validated on a fraction alpha of the Big Data. Each subsample generates a predictive model that is ranked based on performance metrics (e.g., Mean Square Error-MSE and Accuracy) during the first validation step. IMPORTANT - Memory limits to run CBDA: see https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html for various limitations on memory needs while running R under different OS. As far as CBDA is concerned, a CBDA object can be up to 200-300 Mb. The space needed to save all the workspaces however may need to be as large as 1-5 Gb, depending on the number of subsamples. We are working on an new CBDA implementation that reduces the storage constraints.

Value

CBDA object with validation results and 3 RData workspaces

References

See https://github.com/SOCR/CBDA/releases for details on the CBDA protocol and the manuscript "Controlled Feature Selection and Compressive Big Data Analytics: Applications to Big Biomedical and Health Studies" [under review] authored by Simeone Marino, Jiachen Xu, Yi Zhao, Nina Zhou, Yiwang Zhou, Ivo D. Dinov from the University of Michigan

Examples

```
# Installation
# Please upload the Windows binary and/or source CBDA_1.0.0 files from
# the CBDA Github repository https://github.com/SOCR/CBDA/releases
## Not run:
# Installation from the Windows binary (recommended for Windows systems)
install.packages("/filepath/CBDA_1.0.0_binary_Windows.zip", repos = NULL, type = "win.binary")

# Installation from the source (recommended for Macs and Linux systems)
install.packages("/filepath/CBDA_1.0.0_source_.tar.gz", repos = NULL, type = "source")

# Initialization
# This function call installs (if needed) and attaches all the necessary packages to run
```

```
# the CBDA package v1.0.0. It should be run before any production run or test.
# The output shows a table where for each package a TRUE or FALSE is displayed.
# Thus the necessary steps can be pursued in case some package has a FALSE.
CBDA_initialization()

# Set the specs for the synthetic dataset to be tested
n = 300          # number of observations
p = 100          # number of variables

# Generate a nxp matrix of IID variables (e.g., ~N(0,1))
X1 = matrix(rnorm(n*p), nrow=n, ncol=p)

# Setting the nonzero variables - signal variables
nonzero=c(1,100,200,300,400,500,600,700,800,900)

# Set the signal amplitude (for noise level = 1)
amplitude = 10

# Allocate the nonzero coefficients in the correct places
beta = amplitude * (1:p %in% nonzero)

# Generate a linear model with a bias (e.g., white  noise ~N(0,1))
ztemp <- function() X1 %*% beta + rnorm(n)
z = ztemp()

# Pass it through an inv-logit function to
# generate the Bernoulli response variable Ytemp
pr = 1/(1+exp(-z))
Ytemp = rbinom(n,1,pr)
X2 <- cbind(Ytemp,X1)

dataset_file ="Binomial_dataset_3.txt"

# Save the synthetic dataset
a <- tempdir()
write.table(X2, file = paste0(file.path(a),'/',dataset_file), sep=",")

# The file is now stored in the directory a
a
list.files(a)

# Load the Synthetic dataset
Data = read.csv(paste0(file.path(a),'/',dataset_file),header = TRUE)
Ytemp <- Data[,1] # set the outcome
original_names_Data <- names(Data)
cols_to_eliminate=1
Xtemp <- Data[-cols_to_eliminate] # set the matrix X of features/covariates
original_names_Xtemp <- names(Xtemp)

# Add more wrappers/algorithms to the SuperLearner ensemble predictor
# It can be commented out if only the default set of algorithms are used,
# e.g., algorithm_list = c("SL.glm","SL.xgboost","SL.glmnet","SL.svm",
#                          "SL.randomForest","SL.bartMachine")
```

```
# This defines a "new" wrapper, based on the default SL.glmnet
 SL.glmnet.0.75 <- function(..., alpha = 0.75,family="binomial"){
                 SL.glmnet(..., alpha = alpha, family = family)}

 test_example <- c("SL.glmnet","SL.glmnet.0.75")

# Call the CBDA function
# Multicore functionality NOT enabled
CBDA_object <- CBDA.training(Ytemp , Xtemp , M = 12 , Nrow_min = 50, Nrow_max = 70,
              top = 10, max_covs = 8 , min_covs = 3,algorithm_list = test_example ,
              workspace_directory = a)

# Multicore functionality enabled
test_example <- c("SL.xgboost","SL.svm")
CBDA_test <- CBDA.training(Ytemp , Xtemp , M = 40 , Nrow_min = 50, Nrow_max = 70,
               N_cores = 2 , top = 30, max_covs = 20 ,
                min_covs = 5 , algorithm_list = test_example ,
              workspace_directory = a)

## End(Not run)
```

`CBDA_CleanUp` *CBDA Clean up function for Compressive Big Data Analytics*

Description

This CBDA cleans the current directory where all the intermediate workspaces have been created.

Usage

```
CBDA_CleanUp(label = "CBDA_package_test", workspace_directory = tempdir())
```

Arguments

`label`	This is the label appended to RData workspaces generated within the CBDA calls
`workspace_directory`	Directory where the results and workspaces are saved

Value

value

CBDA_Consolidation *CBDA Consolidation function for Compressive Big Data Analytics*

Description

This CBDA function consolidates all the M workspaces generated in the Learning/Training step into a single workspace. It also ranks all the predictive models and selects the **top** ones to be sifted for top predictive features to be passed to the next step (i.e., **the Validation Step**).

Usage

```
CBDA_Consolidation(top, max_covs, M, misValperc, range_k, range_n, label,
  workspace_directory = tempdir())
```

Arguments

top	Top predictions to select out of the M
max_covs	Top features to display and include in the Validation Step where nested models are tested
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
range_k	Features Sampling Range - FSR
range_n	Cases Sampling Range - CSR
label	This is the label appended to RData workspaces generated within the CBDA calls
workspace_directory	Directory where the results and workspaces are saved

Value

value

CBDA_Consolidation.pipeline *CBDA Consolidation function for Compressive Big Data Analytics - LONI pipeline*

Description

This CBDA function consolidates all the M workspaces generated in the Learning/Training step into a single workspace. It also ranks all the predictive models and selects the **top** ones to be sifted for top predictive features to be passed to the next step (i.e., **the Validation Step**).

Usage

```
CBDA_Consolidation.pipeline(top, max_covs, M, misValperc, range_k, range_n,
  label, workspace_directory = tempdir())
```

Arguments

top	Top predictions to select out of the M
max_covs	Top features to display and include in the Validation Step where nested models are tested
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
range_k	Features Sampling Range - FSR
range_n	Cases Sampling Range - CSR
label	This is the label appended to RData workspaces generated within the CBDA calls
workspace_directory	Directory where the results and workspaces are saved

Value

value

CBDA_initialization *CBDA Initialization function for Compressive Big Data Analytics*

Description

This CBDA function installs and attaches all the packages needed to run the CBDA. A user-defined list of packages can be passed as argument. Itis recommended to first execute the function without any arguments.

Usage

```
CBDA_initialization(pkg = c("missForest", "stats", "utils", "prettydoc",
  "foreach", "SuperLearner", "knockoff", "caret", "smotefamily", "parallel",
  "doParallel", "glmnet"), install = FALSE)
```

Arguments

pkg	List of packages to install and attach for running the CBDA algorithm. A default list is already defined.
install	Option to setup installation and attachment of the listed package. Set to FALSE by default

Value

value

`CBDA_spectrum_plots` *CBDA Spectrum plot function for Compressive Big Data Analytics*

Description

This CBDA function generates histograms of the feature counts/densities as returned by the Accuracy and MSE metrics after the Learning/Training step.

Usage

```
CBDA_spectrum_plots(top)
```

Arguments

`top`	Top ranked predictive models from the Learning/Training step

Value

value

`CBDA_Stopping_Criteria`
Stopping Criteria function for Compressive Big Data Analytics

---

Description

This CBDA function generates a stopping criteria for the *max_covs - min_covs* nested predictive models generated in the previous step. It also populates the CBDA object.

Usage

```
CBDA_Stopping_Criteria(label = "CBDA_package_test", Kcol_min = 5,
  Kcol_max = 15, Nrow_min = 30, Nrow_max = 50, misValperc = 0,
  M = 3000, workspace_directory = tempdir(), max_covs = 100,
  min_covs = 5, lambda = 1.005)
```

Arguments

label	This is the label appended to RData workspaces generated within the CBDA calls
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
workspace_directory	Directory where the results and workspaces are saved
max_covs	Top features to include in the Validation Step where nested models are tested
min_covs	Minimum number of top features to include in the initial model for the Validation Step
lambda	Fisher test threshold for MSE (=1.005 by default)

Value

value

CBDA_Stopping_Criteria.pipeline

Stopping Criteria function for Compressive Big Data Analytics

---

Description

This CBDA function generates a stopping criteria for the *max_covs - min_covs* nested predictive models generated in the previous step. It also populates the CBDA object.

Usage

```
CBDA_Stopping_Criteria.pipeline(label = "CBDA_package_test", Kcol_min = 5,
  Kcol_max = 15, Nrow_min = 30, Nrow_max = 50, misValperc = 0,
  M = 3000, workspace_directory = tempdir(), max_covs = 100,
  min_covs = 5, lambda = 1.005)
```

Arguments

label	This is the label appended to RData workspaces generated within the CBDA calls
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
workspace_directory	Directory where the results and workspaces are saved
max_covs	Top features to include in the Validation Step where nested models are tested
min_covs	Minimum number of top features to include in the initial model for the Validation Step
lambda	Fisher test threshold for MSE (=1.005 by default)

Value

value

CBDA_Top_Ranked *CBDA Top-Ranked selection function for Compressive Big Data Analytics*

Description

This CBDA function has all the features of the *Consolidation()* function but allows to choose a different **top** value (i.e., different from the one specified in the main *CBDA()* function

Usage

```
CBDA_Top_Ranked(top_new = 500, label = "CBDA_package_test", Kcol_min = 5,
  Kcol_max = 15, Nrow_min = 30, Nrow_max = 50, misValperc = 0,
  M = 3000, workspace_directory = getwd())
```

Arguments

top_new	The new value for the Top predictions to select out of the M
label	This is the label appended to RData workspaces generated within the CBDA calls
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
workspace_directory	Directory where the results and workspaces are saved

Value

value

CBDA_Validation *CBDA Validation function for Compressive Big Data Analytics*

Description

This CBDA function generates *max_covs - min_covs* nested models based on the ranking returned by the *Consolidation* function. It also consolidates all the *max_covs - min_covs* workspaces into a single one.

Usage

```
CBDA_Validation(label = "CBDA_package_test", alpha = 0.2, Kcol_min = 5,
  Kcol_max = 15, Nrow_min = 30, Nrow_max = 50, misValperc = 0,
  M = 3000, N_cores = 1, top = 1000, workspace_directory = tempdir(),
  max_covs = 100, min_covs = 5)
```

Arguments

label	This is the label appended to RData workspaces generated within the CBDA calls
alpha	Percentage of the Big Data to hold off for Validation
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
N_cores	Number of Cores to use in the parallel implementation
top	Top predictions to select out of the M
workspace_directory	Directory where the results and workspaces are saved
max_covs	Top features to display and include in the Validation Step where nested models are tested
min_covs	Minimum number of top features to include in the initial model for the Validation Step

Value

value

CBDA_Validation.pipeline

CBDA Validation function for Compressive Big Data Analytics - LONI pipeline version

---

Description

This CBDA function generates *max_covs - min_covs* nested models based on the ranking returned by the *Consolidation* function. It also consolidates all the *max_covs - min_covs* workspaces into a single one.

Usage

```
CBDA_Validation.pipeline(job_id_val, Ytemp, Xtemp,
  label = "CBDA_package_test", alpha = 0.2, Kcol_min = 5, Kcol_max = 15,
  Nrow_min = 30, Nrow_max = 50, misValperc = 0, M = 3000, N_cores = 1,
  top = 1000, workspace_directory = tempdir(), max_covs = 100,
  min_covs = 5)
```

Arguments

job_id_val	This is the ID for the job generator in the LONI pipeline interface
Ytemp	This is the output variable (vector) in the original Big Data
Xtemp	This is the input variable (matrix) in the original Big Data
label	This is the label appended to RData workspaces generated within the CBDA calls
alpha	Percentage of the Big Data to hold off for Validation
Kcol_min	Lower bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Kcol_max	Upper bound for the percentage of features-columns sampling (used for the Feature Sampling Range - FSR)
Nrow_min	Lower bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
Nrow_max	Upper bound for the percentage of cases-rows sampling (used for the Case Sampling Range - CSR)
misValperc	Percentage of missing values to introduce in BigData (used just for testing, to mimic real cases).
M	Number of the BigData subsets on which perform Knockoff Filtering and SuperLearner feature mining
N_cores	Number of Cores to use in the parallel implementation
top	Top predictions to select out of the M
workspace_directory	Directory where the results and workspaces are saved
max_covs	Top features to display and include in the Validation Step where nested models are tested
min_covs	Minimum number of top features to include in the initial model for the Validation Step

Value

value

Index