Package ‘heimdall’

March 10, 2026

Title Drift Adaptable Models

Version 1.2.727

Description In streaming data analysis, it is crucial to detect significant shifts in the data distribution or the accuracy of predictive models over time, a phenomenon known as concept drift. The package aims to identify when concept drift occurs and provide methodologies for adapting models in non-stationary environments.
It offers a range of state-of-the-art techniques for detecting concept drift and maintaining model performance. Additionally, the package provides tools for adapting models in response to these changes, ensuring continuous and accurate predictions in dynamic contexts.
Methods for concept drift detection are described in Tavares (2022) <doi:10.1007/s12530-021-09415-z>.

License MIT + file LICENSE

URL https://cefet-rj-dal.github.io/heimdall/,
https://github.com/cefet-rj-dal/heimdall

Encoding UTF-8

RoxygenNote 7.3.3

Imports stats, caret, daltoolbox, ggplot2, Metrics, reticulate, pROC, car

Config/reticulate list(packages = list(list(package = ``scipy"), list(package = ``torch"), list(package = ``pandas"), list(package = ``numpy"), list(package = ``matplotlib"), list(package = ``scikit-learn")))

NeedsCompilation no

Author Lucas Tavares [aut],
Leonardo Carvalho [aut],
Rodrigo Machado [aut],
Diego Carvalho [ctb],
Esther Pacitti [ctb],
Fabio Porto [ctb],
Eduardo Ogasawara [aut, ths, cre] (ORCID: <https://orcid.org/0000-0002-4466-0626>),
CEFET/RJ [cph]

Maintainer Eduardo Ogasawara <eogasawara@ieee.org>

Repository CRAN

Date/Publication 2026-03-10 18:40:02 UTC

Contents

`dfr_adwin` *ADWIN method*

Description

ADWIN (Adaptive Windowing) is a sequential change detector that maintains a variable-length window and tests whether the means of two subwindows differ significantly. In this package, the implementation is primarily used for **virtual concept drift** when it monitors a numeric feature stream, although the same mechanism can also detect **real concept drift** if applied to an error or loss stream. The theoretical basis follows Bifet and Gavaldà (2007) doi:10.1137/1.9781611972771.42.

Usage

```
dfr_adwin(target_feat = NULL, delta = 2e-05)
```

Arguments

`target_feat`	Feature to be monitored.
`delta`	The significance parameter for the ADWIN algorithm.

Value

`dfr_adwin` object

References

Bifet, A., and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443-448. doi:10.1137/1.9781611972771.42

Examples

```
#Use the same example of dfr_cumsum changing the constructor to:
#model <- dfr_adwin(target_feat='serie')
```

`dfr_aedd` *Autoencoder-Based Drift Detection method*

Description

AEDD is an unsupervised multivariate detector that compares reconstruction errors produced by an autoencoder on reference and recent windows. Because it monitors changes in the input distribution rather than classifier performance, this implementation is primarily aimed at **virtual concept drift**. The method follows Kaminskyi, Li, and Muller (2022) doi:10.1109/ICDMW58026.2022.00109.

Usage

```
dfr_aedd(
  encoding_size,
  ae_class = autoenc_encode_decode,
  batch_size = 32,
  num_epochs = 1000,
  learning_rate = 0.001,
  window_size = 100,
  monitoring_step = 1700,
  criteria = "mann_whitney",
  alpha = 0.01,
  reporting = FALSE
)
```

Arguments

`encoding_size`	Encoding Size
`ae_class`	Autoencoder Class
`batch_size`	Batch Size for batch learning
`num_epochs`	Number of Epochs for training
`learning_rate`	Learning Rate
`window_size`	Size of the most recent data to be used
`monitoring_step`	The number of rows that the drifter waits to be is updated
`criteria`	The method to be used to check if there is a drift. May be mann_whitney (default), kolmogorov_smirnov, levene, parametric_threshold, nonparametric_threshold
`alpha`	The significance threshold for the statistical test used in criteria
`reporting`	If TRUE, some data are returned as norm_x_oh, drift_input, hist_proj, and recent_proj.

Value

`dfr_aedd` object

References

Kaminskyi, D., Li, B., and Muller, E. (2022). Reconstruction-based unsupervised drift detection over multivariate streaming data. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. doi:10.1109/ICDMW58026.2022.00109

Examples

```
#See an example of using `dfr_aedd` at this
#https://github.com/cefet-rj-dal/heimdall/blob/main/multivariate/dfr_aedd.md
```

`dfr_cusum` *Cumulative Sum for Concept Drift Detection (CUSUM) method*

Description

CUSUM is a sequential analysis procedure that accumulates deviations in a monitored signal and raises an alarm when the cumulative evidence exceeds a threshold. In this package, the detector is implemented as an error-based monitor, so it is primarily intended for **real concept drift** affecting predictive performance. The concept-drift adaptation follows the sequential change-detection literature discussed by Muthukrishnan, Berg, and Wu (2007) doi:10.1109/ICDMW.2007.89.

Usage

```
dfr_cusum(lambda = 100)
```

Arguments

`lambda`	Necessary level for warning zone (2 standard deviation)

Value

`dfr_cusum` object

References

Muthukrishnan, S., Berg, E., and Wu, Y. (2007). Sequential change detection on data streams. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. doi:10.1109/ICDMW.2007.89

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses an error-based drift detector with a synthetic a
# model residual where 1 is an error and 0 is a correct prediction.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL
data$prediction <- st_drift_examples$univariate$serie > 4

model <- dfr_cusum()

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$prediction)){
 output <- update_state(output$obj, data$prediction[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

`dfr_ddm` *Adapted Drift Detection Method (DDM) method*

Description

DDM monitors the online error rate of a predictive model under the PAC-learning assumption that, in a stationary environment, the error should decrease or remain stable as more samples are observed. Because it operates on the classifier error stream, it is primarily a detector of **real concept drift**. The method follows Gama et al. (2004) doi:10.1007/978-3-540-28645-5_29.

Usage

```
dfr_ddm(min_instances = 30, warning_level = 2, out_control_level = 3)
```

Arguments

`min_instances`	The minimum number of instances before detecting change
`warning_level`	Necessary level for warning zone (2 standard deviation)
`out_control_level`	Necessary level for a positive drift detection

Value

`dfr_ddm` object

References

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence - SBIA 2004*, 286-295. doi:10.1007/978-3-540-28645-5_29

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses an error-based drift detector with a synthetic a
# model residual where 1 is an error and 0 is a correct prediction.

data(st_drift_examples)
```

```
data <- st_drift_examples$univariate
data$event <- NULL
data$prediction <- st_drift_examples$univariate$serie > 4

model <- dfr_ddm()

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$prediction)){
 output <- update_state(output$obj, data$prediction[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

---

dfr_ecdd *Adapted EWMA for Concept Drift Detection (ECDD) method*

Description

ECDD applies an exponentially weighted moving average (EWMA) control chart to the online classification error stream. Since it monitors predictive errors directly, it is primarily designed to detect **real concept drift**. The method follows Ross et al. (2012), who adapted EWMA charts for concept-drift detection in streaming classifiers doi:10.1016/j.patrec.2011.08.019.

Usage

```
dfr_ecdd(lambda = 0.2, min_run_instances = 30, average_run_length = 100)
```

Arguments

`lambda`	EWMA smoothing parameter
`min_run_instances`	The minimum number of instances before detecting change
`average_run_length`	Desired Average Run Length (ARL)

Value

`dfr_ecdd` object

References

Ross, G. J., Adams, N. M., Tasoulis, D. K., and Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2), 191-198. doi:10.1016/j.patrec.2011.08.019

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses an error-based drift detector where 1 is an error and 0 is a correct prediction.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL
data$prediction <- st_drift_examples$univariate$serie > 4

model <- dfr_ecdd()

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$prediction)){
 output <- update_state(output$obj, data$prediction[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

`dfr_eddm` *Adapted Early Drift Detection Method (EDDM) method*

Description

EDDM extends DDM by monitoring the distance between classification errors instead of only the error rate, which makes it more sensitive to gradual degradation. Because it operates on the model error stream, it is primarily intended for **real concept drift**. The method follows Baena-Garcia et al. (2006), who proposed EDDM for improved detection of gradual drift.

Usage

```
dfr_eddm(
  min_instances = 30,
  min_num_errors = 30,
  warning_level = 0.95,
  out_control_level = 0.9
)
```

Arguments

`min_instances`	The minimum number of instances before detecting change
`min_num_errors`	The minimum number of errors before detecting change
`warning_level`	Necessary level for warning zone
`out_control_level`	Necessary level for a positive drift detection

Value

`dfr_eddm` object

References

Baena-Garcia, M., del Campo-Avila, J., Fidalgo, R., Bifet, A., Gavaldà, R., and Morales-Bueno, R. (2006). Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*.

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses an error-based drift detector with a synthetic a
# model residual where 1 is an error and 0 is a correct prediction.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL
data$prediction <- st_drift_examples$univariate$serie > 4

model <- dfr_eddm()

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$prediction)){
 output <- update_state(output$obj, data$prediction[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

dfr_hddm — *Adapted Hoeffding Drift Detection Method (HDDM) method*

Description

HDDM_A is a sequential detector based on Hoeffding's inequality that tests whether the mean of the monitored error stream has increased beyond statistically expected fluctuations. Because this implementation is error-based, it is primarily targeted at **real concept drift**. The theoretical basis follows Frias-Blanco et al. (2015) doi:10.1109/TKDE.2014.2345382.

Usage

```
dfr_hddm(
  drift_confidence = 0.001,
  warning_confidence = 0.005,
  two_side_option = TRUE
)
```

Arguments

`drift_confidence`	Confidence to the drift
`warning_confidence`	Confidence to the warning
`two_side_option`	Option to monitor error increments and decrements (two-sided) or only increments (one-sided)

Value

`dfr_hddm` object

References

Frias-Blanco, I., del Campo-Avila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Diaz, A., and Caballero-Mota, Y. (2015). Online and nonparametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3), 810-823. doi:10.1109/TKDE.2014.2345382

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses an error-based drift detector with a synthetic a
# model residual where 1 is an error and 0 is a correct prediction.

data(st_drift_examples)
```

```
data <- st_drift_examples$univariate
data$event <- NULL
data$prediction <- st_drift_examples$univariate$serie > 4

model <- dfr_hddm()

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$prediction)){
 output <- update_state(output$obj, data$prediction[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

`dfr_inactive` *Inactive dummy detector*

Description

Implements Inactive Dummy Detector

Usage

```
dfr_inactive()
```

Value

Drifter object

Examples

```
# See ?hcd_ddm for an example of DDM drift detector
```

`dfr_kldist` *KL Distance method*

Description

This detector compares consecutive reference and recent windows through the Kullback-Leibler divergence estimated from their empirical distributions. In this package, it is primarily used for **virtual concept drift**, since it monitors changes in the distribution of a numeric feature stream rather than predictive error. The statistical foundation is the Kullback-Leibler divergence introduced by Kullback and Leibler (1951).

Usage

```
dfr_kldist(target_feat = NULL, window_size = 100, p_th = 0.05, data = NULL)
```

Arguments

`target_feat`	Feature to be monitored.
`window_size`	Size of the sliding window
`p_th`	Drift threshold applied to the KL divergence
`data`	Already collected data to avoid cold start.

Value

`dfr_kldist` object

References

Kullback, S., and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1), 79-86. doi:10.1214/aoms/1177729694

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses a dist-based drift detector with a synthetic dataset.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL

model <- dfr_kldist(target_feat='serie')

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$serie)){
 output <- update_state(output$obj, data$serie[i])
```

```
  if (output$drift){
    type <- 'drift'
    output$obj <- reset_state(output$obj)
  }else{
    type <- ''
  }
  detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

dfr_kswin — *KSWIN method*

Description

KSWIN applies a Kolmogorov-Smirnov test between a recent window and a reference sample drawn from older observations. In this package, the method is primarily used for **virtual concept drift**, because it monitors distributional changes in a numeric feature stream. The method follows Raab et al. (2020) doi:10.1016/j.neucom.2019.11.111.

Usage

```
dfr_kswin(
  target_feat = NULL,
  window_size = 1500,
  stat_size = 500,
  alpha = 1e-07,
  data = NULL
)
```

Arguments

target_feat	Feature to be monitored.
window_size	Size of the sliding window (must be > 2*stat_size)
stat_size	Size of the statistic window
alpha	Probability for the test statistic of the Kolmogorov-Smirnov-Test The alpha parameter is very sensitive, therefore should be set below 0.01.
data	Already collected data to avoid cold start.

Value

dfr_kswin object

References

Raab, C., Heusinger, M., and Schleif, F.-M. (2020). Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416, 340-351. doi:10.1016/j.neucom.2019.11.111

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses a dist-based drift detector with a synthetic dataset.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL

model <- dfr_kswin(target_feat='serie')

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$serie)){
 output <- update_state(output$obj, data$serie[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

`dfr_lbdd` *Levene Based Drift Detection Method method*

Description

LBDD is a window-based detector that compares the variability of reference and recent samples using Levene's test. Because it monitors changes in the distribution of an observed feature rather than model performance, it is primarily aimed at **virtual concept drift**. In this package, the detector follows the statistical-testing approach discussed by Giusti et al. (2021) for drift analysis, using Levene's variance test as its core mechanism.

Usage

```
dfr_lbdd(target_feat = NULL, alpha = 0.01, window_size = 1500)
```

Arguments

`target_feat`	Feature to be monitored
`alpha`	Probability theshold for the test statistic
`window_size`	Size of the sliding window

Value

`dfr_lbdd` object

References

Giusti, L., Carvalho, L., Gomes, A. T., Coutinho, R., Soares, J., and Ogasawara, E. (2021). Analysing flight delay under concept drift. *Evolving Systems*. doi:10.1007/s12530-021-09415-z

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses a dist-based drift detector with a synthetic dataset.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL

model <- dfr_lbdd(target_feat='depart_visibility')

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$serie)){
 output <- update_state(output$obj, data$serie[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

`dfr_mcdd` *Mean Comparison Distance method*

Description

MCDD is a window-based detector that compares the location of reference and recent samples by means of hypothesis tests on their central tendency. Because it monitors the distribution of observed features rather than predictive errors, it is primarily intended for **virtual concept drift**. In this package, the detector follows the statistical-testing perspective adopted by Giusti et al. (2021) for drift analysis.

Usage

```
dfr_mcdd(target_feat = NULL, alpha = 1e-08, window_size = 1500)
```

Arguments

`target_feat`	Feature to be monitored
`alpha`	Probability theshold for all test statistics
`window_size`	Size of the sliding window

Value

`dfr_mcdd` object

References

Giusti, L., Carvalho, L., Gomes, A. T., Coutinho, R., Soares, J., and Ogasawara, E. (2021). Analysing flight delay under concept drift. *Evolving Systems*. doi:10.1007/s12530-021-09415-z

Examples

```
library(daltoolbox)
library(heimdall)

# This example uses a dist-based drift detector with a synthetic dataset.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL

model <- dfr_mcdd(target_feat='depart_visibility')

detection <- NULL
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$serie)){
 output <- update_state(output$obj, data$serie[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, data.frame(idx=i, event=output$drift, type=type))
}

detection[detection$type == 'drift',]
```

`dfr_multi_criteria` *Multi Criteria Drifter sub-class*

Description

Implements Multi Criteria drift detectors

Usage

```
dfr_multi_criteria(drifter_list, combination = "or", fuzzy_window = 10)
```

Arguments

`drifter_list`	List of drifters to combine.
`combination`	How the drifters will be combined. Possible values: 'fuzzy', 'or', 'and'.
`fuzzy_window`	Sets the fuzzy window size. Only if combination = 'fuzzy'.

Value

Drifter object

`dfr_page_hinkley` *Adapted Page Hinkley method*

Description

The Page-Hinkley test is a sequential change-point detector that monitors cumulative deviations from a running mean and signals a change when those deviations grow persistently. In this package, the implementation is primarily used for **virtual concept drift** when it monitors a numeric feature stream, although the same statistic can also be applied to error streams to detect **real concept drift**. The method is based on Page (1954) and the later streaming adaptation popularized in data-stream mining.

Usage

```
dfr_page_hinkley(
  target_feat = NULL,
  min_instances = 30,
  delta = 0.005,
  threshold = 50,
  alpha = 1 - 1e-04
)
```

Arguments

`target_feat`	Feature to be monitored.
`min_instances`	The minimum number of instances before detecting change
`delta`	The delta factor for the Page Hinkley test
`threshold`	The change detection threshold (lambda)
`alpha`	The forgetting factor, used to weight the observed value and the mean

Value

`dfr_page_hinkley` object

References

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2), 100-115. doi:10.2307/2333009

Examples

```
library(daltoolbox)
library(heimdall)

# This example assumes a model residual where 1 is an error and 0 is a correct prediction.

data(st_drift_examples)
data <- st_drift_examples$univariate
data$event <- NULL
data$prediction <- st_drift_examples$univariate$serie > 4


model <- dfr_page_hinkley(target_feat='serie')

detection <- c()
output <- list(obj=model, drift=FALSE)
for (i in 1:length(data$serie)){
 output <- update_state(output$obj, data$serie[i])
 if (output$drift){
   type <- 'drift'
   output$obj <- reset_state(output$obj)
 }else{
   type <- ''
 }
 detection <- rbind(detection, list(idx=i, event=output$drift, type=type))
}

detection <- as.data.frame(detection)
detection[detection$type == 'drift',]
```

`dfr_passive` *Passive dummy detector*

Description

Implements Passive Dummy Detector

Usage

```
dfr_passive()
```

Value

Drifter object

Examples

```
# See ?hcd_ddm for an example of DDM drift detector
```

`dist_based` *Distribution Based Drifter sub-class*

Description

Implements Distribution Based drift detectors

Usage

```
dist_based(target_feat)
```

Arguments

`target_feat`	Feature to be monitored.

Value

Drifter object

`drifter` *Drifter*

Description

Ancestor class for drift detection

Usage

```
drifter()
```

Value

Drifter object

Examples

```
# See ?dd_ddm for an example of DDM drift detector
```

`error_based` *Error Based Drifter sub-class*

Description

Implements Error Based drift detectors

Usage

```
error_based()
```

Value

Drifter object

Examples

```
# See ?hcd_ddm for an example of DDM drift detector
```

`fit.drifter` *Process Batch*

Description

Process Batch

Usage

```
## S3 method for class 'drifter'
fit(obj, data, prediction, ...)
```

Arguments

`obj`	Drifter object
`data`	data batch in data frame format
`prediction`	prediction batch as vector format
`...`	opitional arguments

Value

updated Drifter object

`metric` *Metric*

Description

Ancestor class for metric calculation

Usage

```
metric()
```

Value

Metric object

Examples

```
# See ?metric for an example of DDM drift detector
```

`mt_accuracy` *Accuracy Calculator*

Description

Class for accuracy calculation

Usage

```
mt_accuracy()
```

Value

Metric object

Examples

```
# See ?mt_accuracy for an example of Accuracy Calculator
```

`mt_fscore` *FScore Calculator*

Description

Class for FScore calculation

Usage

```
mt_fscore(f = 1)
```

Arguments

`f`	The F parameter for the F-Score metric

Value

Metric object

Examples

```
# See ?mt_fscore for an example of FScore Calculator
```

`mt_precision` *Precision Calculator*

Description

Class for precision calculation

Usage

```
mt_precision()
```

Value

Metric object

Examples

```
# See ?mt_precision for an example of Precision Calculator
```

`mt_recall` *Recall Calculator*

Description

Class for recall calculation

Usage

```
mt_recall()
```

Value

Metric object

Examples

```
# See ?mt_recall for an example of Recall Calculator
```

`mt_rocauc` *ROC AUC Calculator*

Description

Class for QOC AUC calculation

Usage

```
mt_rocauc()
```

Value

Metric object

Examples

```
# See ?mt_rocauc for an example of ROC AUC Calculator
```

`mv_dist_based` *Multivariate Distribution Based Drifter sub-class*

Description

Implements Multivariate Distribution Based drift detectors

Usage

```
mv_dist_based()
```

Value

Drifter object

`norm` *Norm*

Description

Ancestor class for normalization techniques

Usage

```
norm(norm_class)
```

Arguments

`norm_class`	Normalizer class

Value

Norm object

Examples

```
# See ?norm for an example of DDM drift detector
```

`nrm_memory` *Memory Normalizer*

Description

Normalizer that has own memory

Usage

```
nrm_memory(norm_class = minmax())
```

Arguments

`norm_class`	Normalizer class

Value

Norm object

Examples

```
# See ?nrm_mimax for an example of Memory Normalizer
```

`reset_state` *Reset State*

Description

Reset Drifter State

Usage

```
reset_state(obj)
```

Arguments

`obj`	Drifter object

Value

updated Drifter object

Examples

```
# See ?hcd_ddm for an example of DDM drift detector
```

`stealthy` *Stealthy*

Description

Ancestor class for drift adaptive models

Usage

```
stealthy(
  model,
  drift_method,
  monitored_features = NULL,
  norm_class = daltoolbox::zscore(),
  warmup_size = 100,
  th = 0.5,
  target_uni_drifter = FALSE,
  incremental_memory = TRUE,
  verbose = FALSE,
  reporting = FALSE
)
```

Arguments

`model`	The algorithm object to be used for predictions
`drift_method`	The algorithm object to detect drifts
`monitored_features`	List of features that will be monitored by the drifter
`norm_class`	Class used to perform normalization
`warmup_size`	Number of rows used to warmup the drifter. No drift will be detected during this phase
`th`	The threshold to be used with classification algorithms
`target_uni_drifter`	Passes the prediction target to the drifts as the target feat when the drifter is univariate and dist_based.
`incremental_memory`	If true, the model will retrain with all available data whenever the fit is called. If false, it only retrains when a drift is detected.
`verbose`	if TRUE shows drift messages
`reporting`	If TRUE, some data are returned as norm_x_oh, drift_input, hist_proj, and recent_proj.

Value

Stealthy object

Examples

```
# See ?dd_ddm for an example of DDM drift detector
```

`st_drift_examples` *Synthetic time series for concept drift detection*

Description

A list of multivariate time series for drift detection

- example1: a bivariate dataset with one multivariate concept drift example

#'

Usage

```
data(st_drift_examples)
```

Format

A list of time series.

Source

Stealthy package

References

Stealthy package

Examples

```
data(st_drift_examples)
dataset <- st_drift_examples$example1
```

`update_state` *Update State*

Description

Update Drifter State

Usage

```
update_state(obj, value)
```

Arguments

`obj`	Drifter object
`value`	a value that represents a processed batch

Value

updated Drifter object

Examples

```
# See ?hcd_ddm for an example of DDM drift detector
```

Index