Package ‘heimdall’

June 22, 2024

Title Drift Adaptable Models

Version 1.0.707

Description
By analyzing streaming datasets, it is possible to observe significant changes in the data distribution or models' accuracy during their prediction (concept drift). The goal of 'heimdall' is to measure when concept drift occurs. The package makes available several state-of-the-art methods. It also tackles how to adapt models in a nonstationary context. Some concept drifts methods are described in Tavares (2022) <doi:10.1007/s12530-021-09415-z>.

License MIT + file LICENSE

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

Encoding UTF-8

RoxygenNote 7.3.1

Imports stats, daltoolbox, ggplot2, reticulate

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

Depends caret, R (>= 3.5)

NeedsCompilation no

Author Lucas Tavares [aut],
Leonardo Carvalho [aut],
Diego Carvalho [aut],
Esther Pacitti [aut],
Fabio Porto [aut],
Eduardo Ogasawara [aut, ths, cre]
(<`https://orcid.org/0000-0002-0466-0626`>),
Federal Center for Technological Education of Rio de Janeiro (CEFET/RJ)
[cph]

Maintainer Eduardo Ogasawara <`eogasawara@ieee.org`>

Repository CRAN

Date/Publication 2024-06-22 04:20:02 UTC

Contents

dfr_adwin . . . 2
dfr_cumsum . . . 3
dfr_ddm . . . 4
dfr_ecdd . . . 5
dfr_eddm . . . 6
dfr_hddm . . . 7
dfr_kldist . . . 8
dfr_kswin . . . 10
dfr_mcdd . . . 11
dfr_page_hinkley . . . 12
dist_based . . . 13
drifter . . . 14
error_based . . . 14
fit.drifter . . . 15
inactive . . . 15
metric . . . 16
mt_fscore . . . 16
mt_precision . . . 17
mt_recall . . . 17
multi_criteria . . . 18
passive . . . 18
reset_state . . . 19
stealthy . . . 19
st_drift_examples . . . 20
st_real_examples . . . 20
update_state . . . 21

Index **22**

`dfr_adwin` *ADWIN method*

Description

Adaptive Windowing method for concept drift detection doi:10.1137/1.9781611972771.42.

Usage

```
dfr_adwin(target_feat, delta = 0.002)
```

Arguments

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

Value

dfr_adwin object

Examples

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

---

dfr_cumsum — *Cumulative Sum for Concept Drift Detection (CUMSUM) method*

Description

The cumulative sum (CUSUM) is a sequential analysis technique used for change detection.

Usage

```
dfr_cumsum(lambda = 100)
```

Arguments

lambda	Necessary level for warning zone (2 standard deviation)

Value

dfr_cumsum object

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_cumsum()

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

```
 }else{
   type <- ''
 }
 detection <- rbind(detection, list(idx=i, event=output$pred, type=type))
}

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

---

dfr_ddm *Adapted Drift Detection Method (DDM) method*

Description

DDM is a concept change detection method based on the PAC learning model premise, that the learner's error rate will decrease as the number of analysed samples increase, as long as the data distribution is stationary. 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

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_ddm()

detection <- c()
```

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

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

---

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

Description

ECDD is a concept change detection method that uses an exponentially weighted moving average (EWMA) chart to monitor the misclassification rate of an streaming classifier.

Usage

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

Arguments

`lambda`	The minimum number of instances before detecting change
`min_run_instances`	Necessary level for warning zone (2 standard deviation)
`average_run_length`	Necessary level for a positive drift detection

Value

`dfr_ecdd` object

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_ecdd()

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

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

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

Description

EDDM (Early Drift Detection Method) aims to improve the detection rate of gradual concept drift in DDM, while keeping a good performance against abrupt concept drift. doi:2747577a61c70bc3874380130615e15aff76339

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

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_eddm()

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

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

---

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

Description

is a drift detection method based on the Hoeffding's inequality. HDDM_A uses the average as estimator. 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

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_hddm()

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

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

`dfr_kldist` *KL Distance method*

Description

Kullback Leibler Windowing method for concept drift detection.

Usage

```
dfr_kldist(target_feat, window_size = 100, p_th = 0.9, data = NULL)
```

Arguments

`target_feat`	Feature to be monitored.
`window_size`	Size of the sliding window (must be > 2*stat_size)
`p_th`	Probability theshold for the test statistic of the Kullback Leibler distance.
`data`	Already collected data to avoid cold start.

Value

`dfr_kldist` object

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_kldist(target_feat='serie')

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

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

dfr_kswin *KSWIN method*

Description

Kolmogorov-Smirnov Windowing method for concept drift detection doi:10.1016/j.neucom.2019.11.111.

Usage

```
dfr_kswin(
  target_feat,
  window_size = 100,
  stat_size = 30,
  alpha = 0.005,
  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

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_kswin(target_feat='serie')

detection <- c()
```

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

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

---

dfr_mcdd *Mean Comparison Distance method*

Description

Mean Comparison statistical method for concept drift detection.

Usage

```
dfr_mcdd(target_feat, alpha = 0.05, window_size = 100)
```

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

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_mcdd(target_feat='depart_visibility')

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

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

`dfr_page_hinkley` *Adapted Page Hinkley method*

Description

Change-point detection method works by computing the observed values and their mean up to the current moment doi:10.2307/2333009.

Usage

```
dfr_page_hinkley(
  target_feat,
  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

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

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

`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

inactive *Inactive dummy detector*

Description

Implements Inactive Dummy Detector

Usage

```
inactive()
```

Value

Drifter object

Examples

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

metric *Metric*

Description

Ancestor class for metric calculation

Usage

```
metric()
```

Value

Metric object

Examples

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

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_precision 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
```

`multi_criteria` *Multi Criteria Drifter sub-class*

Description

Implements Multi Criteria drift detectors

Usage

```
multi_criteria()
```

Value

Drifter object

`passive` *Passive dummy detector*

Description

Implements Passive Dummy Detector

Usage

```
passive()
```

Value

Drifter object

Examples

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

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, th = 0.5, verbose = FALSE)
```

Arguments

model	The algorithm object to be used for predictions
drift_method	The algorithm object to detect drifts
th	The threshold to be used with classification algorithms
verbose	if TRUE shows drift messages

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
```

st_real_examples *Real time series for concept drift detection*

Description

A list of real multivariate time series for concept drift detection

- bfd1: Brazilian Flights Data 2023

#'

Usage

```
data(st_real_examples)
```

Format

A list of real multivariate time series.

Source

Stealthy package

References

Stealthy package

Examples

```
data(st_real_examples)
dataset <- st_real_examples$bfd1
```

`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

∗ **datasets**
st_drift_examples, 20
st_real_examples, 20

dfr_adwin, 2
dfr_cumsum, 3
dfr_ddm, 4
dfr_ecdd, 5
dfr_eddm, 6
dfr_hddm, 7
dfr_kldist, 8
dfr_kswin, 10
dfr_mcdd, 11
dfr_page_hinkley, 12
dist_based, 13
drifter, 14

error_based, 14

fit.drifter, 15

inactive, 15

metric, 16
mt_fscore, 16
mt_precision, 17
mt_recall, 17
multi_criteria, 18

passive, 18

reset_state, 19

st_drift_examples, 20
st_real_examples, 20
stealthy, 19

update_state, 21