Package 'HVT'

April 25, 2024

Type Package

Date 2024-04-25

Title Constructing Hierarchical Voronoi Tessellations and Overlay Heatmaps for Data Analysis

Version 24.4.1

Description Facilitates building topology preserving maps for rich multivariate data. Credits to Mu Sigma for their continuous support throughout the development of the package.

License Apache License 2.0

Encoding UTF-8

Imports MASS, deldir, grDevices, splancs, conf.design, stats, dplyr, purrr, magrittr, polyclip, ggplot2, tidyr, scales, cluster, reshape2,data.table, ggforce, plyr, rlang, gganimate, gifski, markovchain, methods

Depends R (>= 3.6.0)

BugReports https://github.com/Mu-Sigma/HVT/issues

URL https://github.com/Mu-Sigma/HVT

RoxygenNote 7.3.1

Suggests rmarkdown, testthat, geozoo, plotly, DT, patchwork, sp, Hmisc, gridExtra, gtable, htmlwidgets, installr, skimr, tibble, devtools, tidyverse, DataExplorer, htmltools, corrplot,knitr,kableExtra

VignetteBuilder knitr

NeedsCompilation no

Author Zubin Dowlaty [aut], Mu Sigma, Inc. [cre]

Maintainer ``Mu Sigma, Inc." <ird.experiencelab@mu-sigma.com>

Repository CRAN

Date/Publication 2024-04-25 02:20:00 UTC

R topics documented:

`displayTable` *Table for displaying summary*

Description

main function for displaying summary from model training

Usage

```
displayTable(
  data,
  columnName,
  value,
  tableType = "summary",
  scroll = TRUE,
  limit = 100
)
```

Arguments

`data`	List. Listed object from trainHVT results
`columnName`	Character. Name of the column that needs highlighting.
`value`	Numeric. The value above which highlighted in red or green.
`tableType`	Character. Type of table to generate ('summary', 'compression')
`scroll`	Logical. A value to have scroll or not in the table.
`limit`	Numeric. A value to indicate how many rows to display. Applicable for summary tableType.

Value

A considated table for the training results

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

See Also

trainHVT

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
displayTable(data = hvt.results[[3]]$compression_summary,
columnName = 'percentOfCellsBelowQuantizationErrorThreshold',
value = 0.8, tableType = "compression")
displayTable(data =hvt.results[[3]][['summary']], columnName= 'Quant.Error',
 value = 0.1, tableType = "summary")
```

---

edaPlots *plots for understanding dataset*

Description

main function that gives all the eda plots

Usage

```
edaPlots(df, time_series = FALSE, time_column)
```

Arguments

df	Dataframe. The input dataset, can be a time series too.
time_series	Logical. A value to indicate whether the dataset is time_series or not.
time_column	Character. The name of the time column in the dataset.

Value

Five objects which includes time series plots, data distribution plots, box plots, correlation matrix plot and a descriptive statistics table.

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
edaPlots(dataset, time_series = TRUE, time_column = 'date')
```

---

getTransitionProbability

Creating Transition Probability table

Description

This is the main function to create transition probability table. The transition probability table quantifies the likelihood of transitioning from one state to another. States: The table includes the current states and the possible next states. Probabilities: For each current state, it lists the probability of transitioning to each of the next possible states.

Usage

```
getTransitionProbability(df, cellid_column, time_column)
```

Arguments

df	Data frame. Input dataframe should contain two columns, cell ID from scoreHVT function and timestamp of that dataset.
cellid_column	Character. Name of the column containing cell IDs.
time_column	Character. Name of the column containing timestamps.

Value

Prints and stores a nested list of dataframes with transition probabilities.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>

See Also

trainHVT
scoreHVT

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)
table <- getTransitionProbability(dataset, cellid_column = "cell_id",time_column = "time_stamp")
```

---

plotAnimatedFlowmap *Generating flow maps and animations based on transition probabilities*

Description

This is the main function for generating flow maps and animations based on transition probabilities. Flow maps are a type of data visualization used to represent movements or transitions between different locations or states. They visually connect points to show the direction and volume of movements, such as the transitions in a Hidden Markov Model. These maps help in understanding the dynamics of the system being studied by visually representing the direction and magnitude of flows or transitions.

Usage

```
plotAnimatedFlowmap(
  hvt_model_output,
  transition_probability_df,
  df,
  animation = NULL,
  flow_map = NULL,
  fps_time = 1,
  fps_state = 1,
  time_duration = 2,
  state_duration = 2,
  cellid_column,
  time_column
)
```

Arguments

hvt_model_output	List. Output from a trainHVT function.
transition_probability_df	Data frame. Output Dataframe from getTransitionProbability function
df	Data frame. Input dataframe should contain two columns, cell ID from scoreHVT function and timestamp of that dataset.
animation	Character. Type of animation ('state_based', 'time_based', 'All' or NULL)
flow_map	Character. Type of flow map ('self_state', 'without_self_state', 'All' or NULL)
fps_time	Numeric. A numeric value for the frames per second of the time transition gif.
fps_state	Numeric. A numeric value for the frames per second of the state transition gif.
time_duration	Numeric. A numeric value for the total duration of the time transition gif.
state_duration	Numeric. A numeric value for the total duration of the state transition gif.
cellid_column	Character. Name of the column containing cell IDs.
time_column	Character. Name of the column containing timestamps

Value

A list of plot and gif objects representing flow maps and animations.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>

See Also

trainHVT
scoreHVT
getTransitionProbability

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)
table <- getTransitionProbability(dataset, cellid_column = "cell_id",time_column = "time_stamp")
plots <- plotAnimatedFlowmap(hvt_model_output = hvt.results, transition_probability_df = table,
df = dataset, animation = 'All', flow_map = 'All',fps_time = 1,fps_state =  1,time_duration = 10,
state_duration = 10,cellid_column = "cell_id", time_column = "time_stamp")
```

plotHVT *Plot the hierarchical tessellations.*

Description

This is the main plotting function to construct hierarchical voronoi tessellations in 1D,2D or Interactive surface plot.

Usage

```
plotHVT(
  hvt.results,
  line.width = 0.5,
  color.vec = "black",
  pch1 = 21,
  centroid.size = 1.5,
  title = NULL,
  maxDepth = NULL,
  child.level,
  hmap.cols,
  quant.error.hmap = NULL,
  n_cells.hmap = NULL,
  label.size = 0.5,
  sepration_width = 7,
  layer_opacity = c(0.5, 0.75, 0.99),
  dim_size = 1000,
  plot.type = "2Dhvt",
  cell_id = FALSE
)
```

Arguments

hvt.results	(2DProj/2Dhvt/2Dheatmap/surface_plot) List. A list containing the ouput of trainHVT function which has the details of the tessellations to be plotted.
line.width	(2Dhvt/2Dheatmap) Numeric Vector. A vector indicating the line widths of the tessellation boundaries for each level.
color.vec	(2Dhvt/2Dheatmap) Vector. A vector indicating the colors of the boundaries of the tessellations at each level.
pch1	(2Dhvt/2Dheatmap) Numeric. Symbol type of the centroids of the tessellations (parent levels). Default value is 21.
centroid.size	(2Dhvt/2Dheatmap) Numeric. Size of centroids of first level tessellations.
title	(2Dhvt) String. Set a title for the plot. (default = NULL)
maxDepth	(2Dhvt) Numeric. An integer indicating the number of levels. (default = NULL)
child.level	(2Dheatmap/surface_plot) Numeric. Indicating the level for which the heat map is to be plotted.

hmap.cols	(2Dheatmap/surface_plot) Numeric or Character. The column number of column name from the dataset indicating the variables for which the heat map is to be plotted.
quant.error.hmap	(2Dheatmap) Numeric. A number indicating the quantization error threshold.
n_cells.hmap	(2Dheatmap/surface_plot) Numeric. An integer indicating the number of cells/clusters per hierarchy (level)
label.size	(2Dheatmap) Numeric. The size by which the tessellation labels should be scaled. (default = 0.5)
sepration_width	(surface_plot) Numeric. An integer indicating the width between two Levels
layer_opacity	(surface_plot) Numeric. A vector indicating the opacity of each layer/ level
dim_size	(surface_plot) Numeric. An integer indicating the dimension size used to create the matrix for the plot
plot.type	Character. An option to indicate which type of plot should be generated. Accepted entries are '1D','2Dproj','2Dhvt','2Dheatmap', 'surface_plot'. Default value is '2Dhvt'.
cell_id	(2Dhvt) Logical. To indicate whether the plot should have Cell IDs or not. (default = FALSE)

Value

plot object containing the hvt, heatmap or interactive surface plot for the given trainHVT results.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>

See Also

trainHVT

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")

#change the 'plot.type' argument to '2Dproj' or '2DHVT' to visualise respective plots.
plotHVT(hvt.results, plot.type='1D')

#change the 'plot.type' argument to 'surface_plot' to visualise Interactive surface plot
plotHVT(hvt.results,child.level = 1,
hmap.cols = "DAX", plot.type = '2Dheatmap')
```

plotModelDiagnostics *Make the diagnostic plots for hierarchical voronoi tessellations model.*

Description

This is the main function that generates diagnostic plots for hierarchical Voronoi tessellations models and scoring.

Usage

```
plotModelDiagnostics(model_obj)
```

Arguments

model_obj	List. A list of model_obj obtained from the trainHVT function or scoring object

Value

plot object containing diagnostics plots for the hvt training or scoring For hvt training, Minimum Intra-DataPoint Distance Plot, Minimum Intra-Centroid Distance Plot Mean Absolute Deviation Plot,Distribution of Number of Observations in Cells, for Training Data and Mean Absolute Deviation Plot for Validation Data are plotted. For hvt scoring Mean Absolute Deviation Plot for Training Data and Validation Data are plotted

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>

See Also

plotHVT

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans",diagnose = TRUE,
                        hvt_validation = TRUE)
plotModelDiagnostics(hvt.results)
```

`plotNovelCells` *Plot the identified outlier cell(s) in the voronoi tessellations map.*

Description

This is the main plotting function to construct hierarchical voronoi tessellations and highlight the cells using the compressed HVT map.

Usage

```
plotNovelCells(
  plot.cells,
  hvt.map,
  line.width = c(0.6),
  color.vec = c("#141B41"),
  pch = 21,
  centroid.size = 0.5,
  title = NULL,
  maxDepth = 1
)
```

Arguments

`plot.cells`	Vector. A vector indicating the cells to be highlighted in the map
`hvt.map`	List. A list containing the output of `trainHVT` function which has the details of the tessellations to be plotted
`line.width`	Numeric Vector. A vector indicating the line widths of the tessellation boundaries for each level
`color.vec`	Vector. A vector indicating the colors of the boundaries of the tessellations at each level
`pch`	Numeric. Symbol type of the centroids of the tessellations (parent levels) Default value is 21.
`centroid.size`	Numeric. Size of centroids of first level tessellations. Default value is 0.5
`title`	String. Set a title for the plot. (default = NULL)
`maxDepth`	Numeric. An integer indicating the number of levels. (default = NULL)

Value

Returns a ggplot object containing hierarchical voronoi tessellations plot highlighting the outlier cells in the map

Author(s)

Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>

See Also

trainHVT
plotHVT

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")
#selected 55,58 are for demo purpose
plotNovelCells(c(55,58),hvt.results)
```

---

plotQuantErrorHistogram

Make the diagnostic plots for hierarchical voronoi tessellations model.

Description

This is the function that produces histograms displaying the distribution of Quantized Error (QE) values for both train and test datasets, highlighting mean values with dashed lines for quick evaluation.

Usage

```
plotQuantErrorHistogram(hvt.results, hvt.scoring)
```

Arguments

hvt.results	List. A list of hvt.results obtained from the trainHVT function.
hvt.scoring	List. A list of hvt.scoring obtained from the scoreHVT function.

Value

Returns the ggplot object containing the Quantized Error distribution plots for the given HVT results and scoring

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>

See Also

plotHVT

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
#Split in train and test
train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]
#model training
hvt.results<- trainHVT(train,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE, quant_method = "kmeans")
scoring <- scoreHVT(test, hvt.results)
plotQuantErrorHistogram(hvt.results, scoring)
```

---

plotStateTransition *Creating State Transition Plot*

Description

This is the main function to creating state transition plot from a dataframe. A state transition plot is a type of data visualization used to represent the changes or transitions in states over time for a given system. State refers to a particular condition or status of a cell at a specific point in time. Transition refers to the change of state for a cell from one condition to another over time.

Usage

```
plotStateTransition(
  df,
  sample_size = NULL,
  line_plot = NULL,
  cellid_column,
  time_column
)
```

Arguments

df	Data frame. Input dataframe should contain two columns. Cell ID from scoreHVT function and timestamp of that dataset.
sample_size	Numeric. An integer indicating the Fraction of the dataframe to visualize in the plot. Default value is 0.2
line_plot	Logical. A logical value indicating to create a line plot. Default value is NULL.
cellid_column	Character. Name of the column containing cell IDs.
time_column	Character. Name of the column containing timestamps.

Value

A plotly object representing the state transition plot for the given dataframe.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>

See Also

trainHVT
scoreHVT

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)
plotStateTransition(dataset, sample_size = 1, cellid_column = "cell_id",time_column = "time_stamp")
```

---

reconcileTransitionProbability

Reconciliation of Transition Probability

Description

This is the main function for creating transition probability heatmaps and reconcilation of the same. The Reconciliation of Transition Probability refers to the process of analyzing transition probabilities in a stochastic model like a Markov Chain. It involves ensuring the probabilities accurately reflect real-world dynamics by normalizing them, removing unlikely transitions, and comparing different models. The function creates heatmaps to visually represent these probabilities, aiding in the understanding and analysis of state transitions within the model.

Usage

```
reconcileTransitionProbability(
  df,
  hmap_type = NULL,
  cellid_column,
  time_column
)
```

Arguments

df	Data frame. Input dataframe should contain two columns, cell ID from scoreHVT function and timestamp of that dataset.
hmap_type	Character. Type of heatmap to generate ('self_state', 'without_self_state', or 'All')
cellid_column	Character. Name of the column containing cell IDs.
time_column	Character. Name of the column containing timestamps

Value

A list of plotly heatmap objects representing the transition probability heatmaps.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>

See Also

trainHVT
scoreHVT

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)
reconcileTransitionProbability(dataset, hmap_type = "All",
cellid_column = "cell_id", time_column = "time_stamp")
```

---

removeNovelty *Remove identified outlier cell(s) from the dataset*

Description

This function is used to remove the identified outlier cell(s) from the dataset. It is recommended to run the `trainHVT` function before running this function. It takes input in the form of cell number of the outlier cell(s) identified using the output of the `trainHVT` function and the compressed map (hvt_mapA) generated using the `trainHVT` function. The output of this function is a list of two items: a new map having the data of removed outlier cell(s) and the subset of dataset without outliers.

Usage

```
removeNovelty(outlier_cells, hvt_results)
```

Arguments

`outlier_cells`	Vector. A vector with the cell number of the identified outliers
`hvt_results`	List. A list having the results of the compressed map i.e. output of `trainHVT` function

Value

A list of two items: a map having the data of removed outlier cells and the subset of the dataset without outlier(s) which has to be passed as input argument to `trainHVT` function to generate another map

`[[1]]`	Dataframe. Information about the removed outlier cell(s)
`[[2]]`	Dataframe. Subset of dataset without the outlier cell(s)

Author(s)

Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>

See Also

`trainHVT`
`scoreLayeredHVT`

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")
identified_Novelty_cells <<- c(2, 10)
output_list <- removeNovelty(identified_Novelty_cells, hvt.results)
data_with_novelty <- output_list[[1]]
data_without_novelty <- output_list[[2]]
```

scoreHVT *Score which cell each point in the test dataset belongs to.*

Description

This is the function scores the cell for each point in the test dataset based on a trained hierarchical Voronoi tessellations model. It provides scored data, plots, and mean absolute deviation plots, aiding in the evaluation of model performance.

Usage

```
scoreHVT(
  data,
  hvt.results.model,
  child.level = 1,
  mad.threshold = 0.2,
  line.width = c(0.6, 0.4, 0.2),
  color.vec = c("navyblue", "slateblue", "lavender"),
  normalize = TRUE,
  seed = 300,
  distance_metric = "L1_Norm",
  error_metric = "max",
  yVar = NULL
)
```

Arguments

data	Dataframe. A dataframe containing the test dataset. The dataframe should have all the variable(features) used for training.
hvt.results.model	List. A list obtained from the trainHVT function while performing hierarchical vector quantization on training data. This list provides an overview of the hierarchical vector quantized data, including diagnostics, tessellation details, Sammon's projection coordinates, and model input information.
child.level	Numeric. A number indicating the depth for which the heat map is to be plotted. Each depth represents a different level of clustering or partitioning of the data.
mad.threshold	Numeric. A numeric value indicating the permissible Mean Absolute Deviation which is obtained from Minimum Intra centroid plot of diagnostics.
line.width	Vector. A vector indicating the line widths of the tessellation boundaries for each layer.
color.vec	Vector. A vector indicating the colors of the tessellations boundaries at each layer.
normalize	Logical. A logical value indicating if the dataset should be normalized. When set to TRUE, the data (testing dataset) is standardized by 'mean' and 'sd' of the training dataset referred from the trainHVT(). When set to FALSE, the data is used as such without any changes.

seed	Numeric. Random Seed.
distance_metric	Character. The distance metric can be 'L1_Norm'(Manhattan) or 'L2_Norm'(Eucledian). 'L1_Norm' is selected by default. The distance metric is used to calculate the distance between an 'n' dimensional point and centroid. The distance metric can be different from the one used during training.
error_metric	Character. The error metric can be 'mean' or 'max'. 'max' is selected by default. 'max' will return the max of 'm' values and 'mean' will take mean of 'm' values where each value is a distance between a point and centroid of the cell. The error metric can be different from the one used during training.
yVar	Character. A character or a vector representing the name of the dependent variable(s)

Value

Dataframe containing scored data, plots and mean absolute deviation plots

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>

See Also

trainHVT
plotHVT

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
# Split in train and test
train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]
#model training
hvt.results<- trainHVT(train,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(test, hvt.results)
data_scored <- scoring$scoredPredictedData
```

`scoreLayeredHVT` *Score which cell and what layer each point in the test dataset belongs to*

Description

This is a function that scores the cell and corresponding layer for each point in a test dataset using three hierarchical vector quantization (HVT) models (Map A, Map B, Map C) and returns a dataframe containing the scored layer output. The function incorporates the scored results from each map and merges them to provide a comprehensive result.

Usage

```
scoreLayeredHVT(
  data,
  hvt_mapA,
  hvt_mapB,
  hvt_mapC,
  mad.threshold = 0.2,
  normalize = TRUE,
  seed = 300,
  distance_metric = "L1_Norm",
  error_metric = "max",
  child.level = 1,
  yVar = NULL
)
```

Arguments

`data`	Data Frame. A dataframe containing test dataset. The dataframe should have all the variable(features) used for training.
`hvt_mapA`	A list of hvt.results.model obtained from trainHVT function while performing hierarchical vector quantization on train data
`hvt_mapB`	A list of hvt.results.model obtained from trainHVT function while performing hierarchical vector quantization on data with novelty(s)
`hvt_mapC`	A list of hvt.results.model obtained from trainHVT function while performing hierarchical vector quantization on data without novelty(s)
`mad.threshold`	Numeric. A number indicating the permissible Mean Absolute Deviation
`normalize`	Logical. A logical value indicating if the dataset should be normalized. When set to TRUE, the data (testing dataset) is standardized by mean and sd of the training dataset referred from the trainHVT(). When set to FALSE, the data is used as such without any changes. (Default value is TRUE).
`seed`	Numeric. Random Seed.

distance_metric
: Character. The distance metric can be 'L1_Norm'(Manhattan) or 'L2_Norm'(Eucledian). 'L1_Norm' is selected by default. The distance metric is used to calculate the distance between an 'n' dimensional point and centroid. The distance metric can be different from the one used during training.

error_metric
: Character. The error metric can be 'mean' or 'max'. 'max' is selected by default. 'max' will return the max of 'm' values and 'mean' will take mean of 'm' values where each value is a distance between a point and centroid of the cell. The error metric can be different from the one used during training.

child.level
: Numeric. A number indicating the level for which the heat map is to be plotted.

yVar
: Character. A character or a vector representing the name of the dependent variable(s)

Value

Dataframe containing scored layer output

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>, Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>,Somya Shambhawi <somya.shambhawi@mu-sigma.com>

See Also

trainHVT
plotHVT

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date

train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]

###MAP-A
hvt_mapA <- trainHVT(train, n_cells = 150, depth = 1, quant.err = 0.1,
                     distance_metric = "L1_Norm", error_metric = "max",
                     normalize = TRUE,quant_method = "kmeans")

identified_Novelty_cells <- c(127,55,83,61,44,35,27,77)
output_list <- removeNovelty(identified_Novelty_cells, hvt_mapA)
data_with_novelty <- output_list[[1]]
data_with_novelty <- data_with_novelty[, -c(1,2)]
```

```
### MAP-B
hvt_mapB <- trainHVT(data_with_novelty,n_cells = 10, depth = 1, quant.err = 0.1,
                     distance_metric = "L1_Norm", error_metric = "max",
                     normalize = TRUE,quant_method = "kmeans")
data_without_novelty <- output_list[[2]]

### MAP-C
hvt_mapC <- trainHVT(data_without_novelty,n_cells = 135,
                     depth = 1, quant.err = 0.1, distance_metric = "L1_Norm",
                     error_metric = "max", quant_method = "kmeans",
                     normalize = TRUE)

##SCORE LAYERED
data_scored <- scoreLayeredHVT(test, hvt_mapA, hvt_mapB, hvt_mapC)
```

`trainHVT` *Constructing Hierarchical Voronoi Tessellations*

Description

This is the main function to construct hierarchical voronoi tessellations. The hvq script is used in this function. The output of hvq is hierarchical clustered data which will be the input for constructing tessellations. The data is then represented in 2D coordinates and the tessellations are plotted using these coordinates as centroids. For subsequent levels, transformation is performed on the 2D coordinates to get all the points within its parent tile. Tessellations are plotted using these transformed points as centroids. The lines in the tessellations are chopped in places so that they do not protrude outside the parent polygon. This is done for all the subsequent levels.

Usage

```
trainHVT(
  dataset,
  min_compression_perc = NA,
  n_cells = NA,
  depth = 1,
  quant.err = 0.2,
  projection.scale = 10,
  normalize = FALSE,
  seed = 279,
  distance_metric = c("L1_Norm", "L2_Norm"),
  error_metric = c("mean", "max"),
  quant_method = c("kmeans", "kmedoids"),
  scale_summary = NA,
  diagnose = FALSE,
  hvt_validation = FALSE,
  train_validation_split_ratio = 0.8
)
```

Arguments

dataset	Dataframe. A dataframe, with numeric columns (features) that will be used for training the model.
min_compression_perc	Numeric. An integer, indicating the minimum compression percentage to be achieved for the dataset. It indicates the desired level of reduction in dataset size compared to its original size.
n_cells	Numeric. An integer, indicating the number of cells per hierarchy (level). This parameter determines the granularity or level of detail in the hierarchical vector quantization.
depth	Numeric. An integer, indicating the number of levels. A depth of 1 means no hierarchy (single level), while higher values indicate multiple levels (hierarchy).
quant.err	Numeric. An number indicating the quantization error threshold. A cell will only breakdown into further cells if the quantization error of the cell is above the defined quantization error threshold.
projection.scale	Numeric. A number indicating the scale factor for the tesselations so as to visualize the sub-tesselations well enough. It helps in adjusting the visual representation of the hierarchy to make the sub-tesselations more visible.
normalize	Logical. A logical value indicating if the dataset should be normalized. When set to TRUE, scales the values of all features to have a mean of 0 and a standard deviation of 1 (Z-score).
seed	Numeric. A Random Numeric Seed to preserve the repeatability.
distance_metric	Character. The distance metric can be L1_Norm(Manhattan) or L2_Norm(Eucledian). L1_Norm is selected by default. The distance metric is used to calculate the distance between an n dimensional point and centroid.
error_metric	Character. The error metric can be mean or max. max is selected by default. max will return the max of m values and mean will take mean of m values where each value is a distance between a point and centroid of the cell.
quant_method	Character. The quantization method can be kmeans or kmedoids. Kmeans uses means (centroids) as cluster centers while Kmedoids uses actual data points (medoids) as cluster centers. kmeans is selected by default.
scale_summary	List. A list with user defined mean and standard deviation values for all the features in the dataset. Pass the scale summary when normalize is set to FALSE.
diagnose	Logical. A logical value indicating whether user wants to perform diagnostics on the model. Default value is FALSE.
hvt_validation	Logical. A logical value indicating whether user wants to holdout a validation set and find mean absolute deviation of the validation points from the centroid. Default value is FALSE.
train_validation_split_ratio	Numeric. A numeric value indicating train validation split ratio. This argument is only used when hvt_validation has been set to TRUE. Default value for the argument is 0.8

Value

A Nested list that contains the hierarchical tessellation information. This list has to be given as input argument to plot the tessellations.

[[1]]	A list containing information related to plotting tessellations. This information will include coordinates, boundaries, and other details necessary for visualizing the tessellations
[[2]]	A list containing information related to Sammon's projection coordinates of the data points in the reduced-dimensional space.
[[3]]	A list containing detailed information about the hierarchical vector quantized data along with a summary section containing no of points, Quantization Error and the centroids for each cell.
[[4]]	A list that contains all the diagnostics information of the model when diagnose is set to TRUE. Otherwise NA.
[[5]]	A list that contains all the information required to generates a Mean Absolute Deviation (MAD) plot, if hvt_validation is set to TRUE. Otherwise NA
[[6]]	A list (model info) that contains model generated timestamp, input parameters passed to the model and the validation results.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>, Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>

See Also

plotHVT

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")
```

Index