Internship_Levavasseur_Yann Documentation

Version 1.0

LEVAVASSEUR_Yann

août 22, 2018

Table des matières

1	NoteBooks	3
2	utils	5
3	Installation	17
	Index des modules Python	19

Rendu du travail **Python** effectué dans le cadre d'un stage de fin d'année en Master informatique. **Data mining** sur des données hydrogéologiques, plusieurs techniques sont mises en place dans ce projet principalement de classification non supervisée.

Auteur LEVAVASSEUR Yann

Version 1.0 de 2018

Documentation :

CHAPITRE 1

NoteBooks

Description des fichiers principaux **Notebook.ipynb**

- **Cluster_engine** : Permet d'entrer les hypers parametres necessaires au lancement d'un clustering (ici Kmean ou Kshape) et le stockage de celui-ci dans un fichier Pickle
- **Read_cluster** : Permet d'ouvrir un fichier Pickle d'un Clustering sauvegarder avec de l'analyser avec plusieurs outils visuelles
- **Read_data** : Permet d'afficher et de jouer avec les donnees passee en parametres

2.1 utils package

2.1.1 Submodules

2.1.2 utils.cluster_ts module

class `utils.cluster_ts.ClusterTs` (*ss*)

Bases : `object`

Classe disposant des methodes de transformations et de manipulations des donnees a des fins de partitionnements

classe mere de :

- `kmean`
- `kshape`

Parameters :

- `ss` [`SeriesSupp`] instance du manager de series temporelles

Variables :

- `ts` : `Array[[[float]]]` les series temporelle au format desiree pour le clustering
- `ts_clust` : `Array[int]` Chaque entier est selon son index le cluster auquel appartient l'index referant de *ts*
- `ts_name` : `Array[String]` Nom de la serie temporelle, du capteur a sa granularite (annee, mois, semaine)
- `ss` : `SeriesSupp` instance du manager de series temporelles
- `sampler` : `int` Taille du sampling `sampler()`
- `ploter` : `:class :Plot` Instance d'un objet d'affichage
- `n` : `int` Nombre de cluster
- `capteurs_names` : `Array[String]` Nom de la serie temporelle, du capteur a sa granularite (annee, mois, semaine) *Bientot supprime*
- `from_save` : `Bool` True si les infos sont recuperees d'un cluster sauvegarde
- `proto` : `Array[[[float]]]` Prototype de chaque cluster
- `last_readed` : `{Dict}` Informations recuperer depuis le fichier "Pickle" sauvegarde du cluster etudier
- `store_path` : `String` Chemin vers le dossier de stockage des sauvegardes. N'est plus utilise depuis l'implmentation d'une boite de dialogue pour la recherche de fichier de sauvegarde

- **name_file** : **String** Chemin absolue vers fichier “Pickle”
- **clust_name** : **String** Nom de la technique de clustering de l’instance
- **metric** : **String** Nom de la technique de calcul de distance de l’instance
- **geo** : **class :Geo** Instance Geo
- **cluster_by_name** : **{Dict}** Clustering des series temporelles uniquement par le nom des capteurs sans redondance
- **cluster_by_fullname** : **{Dict}** Clustering des series temporelles uniquement par le nom des capteurs et leurs granularite
- **size_min** : **int** Taille minimale d’une serie pour etre garde lors du preprocessing
- **nb_captteur** : **{Dict}** Clustering des series temporelles uniquement par le nom des capteurs redondance
- **nb_week** : **{Dict}** Lors d’un decoupage en semaine, represente la redondance par capteur des semaines

Example : See : Cluster_engine.ipynb

Notes :

Dependencies :

- tslearn
- pandas
- Pickle

aff_color ()

Affiche les couleurs utilise dans le clustering

captteur_parser ()

Parser des noms de capteurs, pour pouvoir garder en memoire les nom des capteur et leur extension de date selon la TS

Parameters : NA

Returns : NA

check_equal (*iterator*)

Verifie si la TS reste tout le temps sur la meme valeur

Parameters :

- **iterator** : **iterator** la TS

Returns : Bool

clust_hoverview (*n*)

Affiche les TS d’un cluster **n** donnee

Parameters :

- **n** : **int** cluster selectionne

Returns : NA

clust_hoverview_rng (*n*)

DEPRECATED : Tire une TS random d’un cluster **n** pour se donner une idee des membres de ce dernier

Parameters :

- **n** : **int** Le cluster numero n

Returns : NA

get_captor_distribution_in_cluster ()

Retourne le nombre d’occurance des cpteur dans chacun des clusters

Parameters : NA

Returns :

unnamed : **DataFrame** Tableau d’occurance

get_clust_part_for_captor (*cpt*)

recupere les cluster pour capteur **cpt** donne et leur distribution au sein des cluster

Parameters :

- **cpt** : **String** Capteur target

Returns :
res : String Seulement les noms des date pour chacun des clusters

get_cluster_n (*n*)
 Retourne les TS d'un cluster **n**

Parameters :
 — **n : int** Numero de cluster souhaite

Returns :
res : Array[float] Ensemble des TS du cluster

get_part_of_ts (*data, elmt*)
 Selon les Parameters d'elmt retrouve une partie des donnes depuis data

Parameters :
 — **data : {Dict}** Donnee depuis les quelles on souhaite recuperer une partie precise
 — **elmt : {Dict}** Information liee a la demande (date)

Returns :
res_ts : Array[float] TS souhaitee

get_ts_by_captor (*cpt*)
 recupere les TS pour capteur **cpt** donne et leur distribution au sein des cluster

Parameters :
 — **cpt : String** Capteur target

Returns :
res : tuple(String, {Dict}) String represente le capteur et le dictionnaires la distribution des sous TS dans chaque clusters (key = cluster)

highlight_max (*s*)
 Parametre d'affichage surligne les max par ligne de DataFrame

Parameters :
 — **s : pandas** Ligne du tableau

Returns :
unnamed : pandas.style Affichage des max

parse_captor_split (*elmt*)
 Recupere les information d'une TS depuis son nom comme le nom de son capteur et la date

Parameters :
 — **elmt : String** Nom du capteur avec info

Returns :
res : {Dict} Les infos decoupees et range dans un dico

read_cluster (*path=""*)
 Ouvre et recupere toutes les informations d'un fichier pickle(sauvegarde d'un clustering) et update les variable de l'instance pour correspondre

Parameters :
 — **path : String** Chemin d'acces au fichier

Returns : NA

set_size_min (*size*)
 Set taille minimale d'une TS pour etre gardee

Parameters :
 — **size : int** Taille minimale

Returns : NA

show_info ()
 Affiche les informations recuperees depuis le txt d'info de la sauvegarde cluster lie a l'instance

Parameters : NA

Returns : NA

store_cluster (*name*)

Sauvegarde sur forme de fichier pickle associe a un txt d'information la partitionnement actuelle de l'instance

Parameters :

— **name : String** Nom du fichier, represente les parametre principaux de la partitionnement

Returns : NA

style_df (*opt, t*)

tslearn_format_export (*other_data=None*)

Export la variable data vers le format utilise par tslearn pour la partitionnements

Parameters : NA

Returns : NA

2.1.3 utils.data_factory module

class `utils.data_factory.DataFactory` (*cwd*)

Bases : `object`

Factory fournis et importe les donnees en gerant le nombre d'instances

Parameters :

— **cwd : String** chemin d'accès ou le main est exécuté

Variables :

— **path_RG24 : String** Chemin pour récupérer les données RG24

— **path_RG1 : String** Chemin pour récupérer les données RG1

— **path_GW : String** Chemin pour récupérer les données GW

— **RG24 : {Dict}** Les données RG24

— **RG1 : {Dict}** Les données RG1

— **GW : {Dict}** Les données GW

Notes :

— RG24 : Rain gauge, precipitation de pluie journaliere

— RG1 : Rain gauge, precipitation de pluie horaire

— GW : Grand Water, données piezometriques

get_GW ()

Retourne un [DICT] en guise de dataset instancie une seule fois le dataset

get_RG1 ()

Retourne un [DICT] en guise de dataset instancie une seule fois le dataset

get_RG24 ()

Retourne un [DICT] en guise de dataset instancie une seule fois le dataset

get_data (*name*)

Dirige l'importation vers la bonne methode selon le parametre

Parameters :

— **name : String** Nom du type de données a récupérer

Returns : None

get_dataset (*source*)

Recupere et retourne un Dictionnaire de Dataframe importe de multiple fichiers csv

Parameters :

— **source : String** Chemin vers les fichiers csv

Returns :

```

dataset : {Dict} Dictionnaire des donnees par capteurs
path_GW = 'csv_prepro\\GW'
path_RG1 = 'csv_prepro\\RG\\precipitation_1h_RG'
path_RG24 = 'csv_prepro\\RG\\precipitation_RG'

```

2.1.4 utils.file_dialog module

2.1.5 utils.geo module

class `utils.geo.Geo` (*cwd*)

Bases : `object`

Classe entierement dedie a l'affichage des capteur dans l'espace les donnees sont normees suivant un type de geolocalisation geologique propre a la NC, la classe dispose de plusieurs type d'affichage

Parameters :

— **cwd : String** Chemin vers le main

Variables :

- **path_geo_RG : String** Chemin pour recuperer les donnees RG
- **path_geo_GW : String** Chemin pour recuperer les donnees GW
- **geo_RG : {Dict}** Les donnees geo_RG
- **geo_GW : {Dict}** Les donnees geo_GW
- **dist_mat : pandas.DataFrame** Matrice des distances entre les pluviometre et des piezometres

Notes :

- RG : Rain gauge, precipitation de pluie journaliere
- GW : Grand Water, donnees piezometric

distance_dict ()

distance_matrix ()

Retourne une dataframe representant les distances entre piezo et pluvio ainsi que les distances les plus courtes

Parameters : NA

Returns :

gdist_style : pandas.style Matrice stylise des distances

get_geo_data (*source*)

Recupere les donnees geographique

Parameters :

— **source : String** location ou recuperer ces donnees

Returns :

data : pandas.DataFrame Donnees sous forme de tableau

get_minrange_rg (*cpt*)

highlight_min (*s*)

Parametre d'affichage surligne les min par ligne de DataFrame

Parameters :

— **s : pandas** Ligne du tableau

Returns :

unnamed : pandas.style Affichage des min

plot_2D ()

plot_3D ()

Afficha 3D des geolocalisations des ouvrages

Parameters : NA

Returns : NA

`plot_3D_for_all_cluster` (*names*)

`plot_3D_for_one_cluster` (*names*)

`plotly_3D` (*names*)

Afficha 3D des geolocalisations des ouvrages avec plotly cette fois Plus utilise que le premier

Parameters :

— **names : String** Nom des capteurs pour les afficher au survol

Returns : NA

2.1.6 utils.k_mean module

class `utils.k_mean.Kmean` (*ss*)

Bases : `utils.cluster_ts.ClusterTs`

Classe de partitionnement des donnees avec l'algorithm K-mean

Parameters :

— *ss* [SeriesSupp] instance du manager de series temporelles

Variables :

— **seed : int** Valeur d'initialisation de l'algo, random.

— **counter : Counter** repartition des objets au sein des clusters

— **km : TimeSeriesKMeans** Instance de l'algo

— **clust_name : String** Nom de l'algo(affichage des plots)

— **metric : String** Choix du metrics utilise, principalement softdtw ici car tres efficace et rapide

`cluster_counter` ()

Compte les objets au sein des clusters

Parameters : NA

Returns : NA

`k_fit` ()

Effectue le partitionnement

Parameters : NA

Returns : NA

`k_init` (*v=True*)

initialisation de l'instance de l'algorithm avec les parametres actuels

Parameters :

— **v : boolean** Verbose, affiche les info lie au partitionnement

Returns : NA

2.1.7 utils.k_shape module

class `utils.k_shape.Kshape` (*ss*)

Bases : `utils.cluster_ts.ClusterTs`

Classe de partitionnement des donnees avec l'algorithm K-shape

Parameters :

— *ss* [SeriesSupp] instance du manager de series temporelles

Variables :

— **seed : int** Valeur d'initialisation de l'algo, random.

— **counter : Counter** repartition des objets au sein des clusters

- **km : TimeSeriesKMeans** Instance de l’algo
- **clust_name : String** Nom de l’algo(affichage des plots)
- **metric : String** Choix du metrics utilise, principalement softdtw ici car tres efficace et rapide

cluster_counter ()

Compte les objets au sein des clusters

Parameters : NA

Returns : NA

k_fit ()

Effectue le partitionnement

Parameters : NA

Returns : NA

k_init (*v=True*)

initialisation de l’instance de l’algorithm avec les parametres actuels

Parameters :

- **v : boolean** Verbose, affiche les info lie au partitionnement

Returns : NA

2.1.8 utils.parser module

class `utils.parser.Parser`

Bases : `object`

Transforme les donnees d’un format a un autre a l’aide de methodes de transformations

sax_to_spmf (*data, separator=False*)

Du format SAX au format SPMF(un soft de data mining avec son propre format)

Parameters :

- **data : SAX**

Returns : `res_str : String`

2.1.9 utils.plot module

class `utils.plot.Plot` (*cluster*)

Bases : `object`

Classe permetant l’affichage sous forme de graph de donnees

Parameters :

- **cluster : ClusterTs** instance de partitionnement

Variables :

- **mode : String** Type d’affichage des graphes
- **colors : {Dict}** Dictionnaire de couleur pour les plots

change_mode (*m*)

Change le type d’affichage :

- 1 : Markers
- 2 : Lines
- 3 : Lines+Markers

Parameters :

- **m : int [1 :3]** L’entier qui determiner le mode actuel, si en dehors de [1 :3] reste au meme mode qu’avant l’appel

Returns : NA

plot_captor_cluster_cover (*data*)

plot_cluster ()
 Methode d’affichage graphique des clusters formes par l’instance ClusterTs Affiche chaque groupe avec une difference de couleur pour le prototype

plot_cluster_light ()
 Version legere d’affichage pour limiter la consommation de puissance d’un affichage dynamique

plot_histo (*n*)
 Affiche par cluster le nombre d’occurrence des Capteurs de et la granularite maximale

plot_histo_2 (*n*)

plot_prefixspan (*l*)

plot_scatter (*data*)
 Affiche des donnees avant cluster, permet de visualiser les donnees decoupe par exemple data : [DICT] les donnees key : Capteur value : Dataframe

plot_scatter_by_captteur (*data*, *capteur*)

plot_scatter_by_captteur_color_cluster (*data*, *dict_clust*, *rg=None*)
 Affiche un capteur et colorise selon l’appartenance au cluster

plot_scatter_light (*data*)
 Encore une version plus legere de la methode de base data : [DICT] les donnees key : Capteur value : Dataframe

plot_simple_TS (*ts*)
 Plot une liste de valeur

Parameters :
 — **ts : Array[int]** Liste de valeur a afficher

Returns : NA

plot_simple_list (*list*)

2.1.10 utils.prefix_span module

class `utils.prefix_span.PrefixSpanManager` (*sax_engine*, *export=True*)
 Bases : `object`
 Classe d’outil a l’utilisation de prefixspan

Parameters :
 — **sax_engine : SaxEngine** Instance de preprocessing SAX
 — **export : Boolean** Si oui ou non les donnees sont deja exportees au bon format

Variables :
 — **se_instance : SaxEngine** L’instance de class SAX
 — **data : Array[]** Les donnees au format SAX

export_format ()
 Modifie le format pour correspondre au besoin de l’instance de PrefixSpan

frequent (*n*)
 Retourne les frequent de support n

Parameters :
 — **n : int** Support minimal

Returns : Liste des motifs de support minimal n

plot (*l*)

run ()
 Creer l’instance PrefixSpan avec les donnees pretraites

topk (*n*, *c=True*)
 Affiche les motifs les plus frequents(plus grand support) et par default les fermes

Parameters :

— **n : int** Nombre de motifs a afficher

Returns : Liste de motifs frequent

2.1.11 utils.pygapbide module

class `utils.pygapbide.Gapbide` (*sdb, sup, m, n*)

Bases : `object`

backward_check (*pattern, sup, pdb*)

forward_check (*pattern, sup, pdb*)

gen_ll_patterns ()

generate length-l patterns

output (*pattern, sup, pdb*)

override this function to output patterns to files.

run ()

span (*pattern, sup, pdb*)

class `utils.pygapbide.GapbideManager` (*gp=None*)

Bases : `object`

Manager d'une instance : `class :Gapbide`

Variables d'Instance :

— **gp : Gapbide** Instance de l'algo a manager

— **dataset : Array[int]** Liste des donnees sax les entiers font office de symboles

— **res : Array[[int : X*, int : Y, Z(int : Z1, int : Z2, int : Z3)]**

— X* Pattern d'entiers ex : 244695

— Y Support du pattern

— **Z(* Z1 : Position du patterns, indice de la transaction**

— Z2 : Indice au sein de la transaction du debut du patterns

— Z3 : Indice de fin...)

clean_import_db (*db*)

reset_gp (*sup, m, n*)

run ()

2.1.12 utils.sax_engine module

class `utils.sax_engine.SaxEngine` (*nsy, nse*)

Bases : `object`

Manager SAX(SymbolicAggregateApproximation) pour formater les donnees temporelles vers unhe suites de segments a symboles representatifs

Parameters :

— **nsy [int]** Nombre de symbols

— **nsy [int]** Nombre de segments

Variables :

— **raw_data [ArrayList]** Donnees brutes

— **process_data [ArrayList]** Donnees retravaillees pour correspondre au format fit()

— **sax_data_inv [ArrayList]** Donnees au format SAX mais de meme taille len() que les donnees brutes

— **sax_data [ArrayList]** Donnees au format SAX

Example : See : `Sequential_pattern_mining.ipynb`

export_format ()

```

fit (data, export=True)
fit_run (data)
reset ()
    Reinitialise les variables d'instance
    Parameters : NA
    Returns : NA
run ()
set_nb_segment (nb)
set_nb_symbol (nb)
step_run (data)
    
```

2.1.13 utils.series_supp module

```

class utils.series_supp.SeriesSupp (cwd, factory, dataset_name)
    
```

Bases : `object`

Permet d'organiser et de manipuler les données.

Parameters :

- **cwd** : **String** chemin d'accès où le main est exécuté
- **factory** : **Factory** Instance de la factory
- **dataset_name** : **String** Définie le type de données à récupérer

Variables :

- **dataset** : **{Dict}** Le dataset original sans modification
- **tmp_dataset** : **{Dict}** Le dataset actuel avec les modifications
- **years** : **[ARRAY<STRING>]** Setup de découpage par années
- **months** : **[ARRAY<STRING>]** Setup de découpage par mois
- **days** : **[ARRAY<STRING>]** Setup de découpage par années | **[BOOL]** découpage semaines
- **factory** : **DataFactory** Instance de la Factory
- **dataset_name** : **String** Permet de connaître la source souhaitée e.g `import_dataset()`

```

dict_norm ()
    
```

Normalise un dictionnaire de TS

```

dict_round ()
    
```

Normalise un dictionnaire de TS

```

dict_smooth (wind=24, col='Valeur')
    
```

Dictionnaire full smooth

Parameters :

- **wind** : **int** Taille de la fenêtre 24 pour smooth journalier
- **col** : **str** (DEFAUT = « Valeur ») Colonne cible de l'action

Returns : NA

```

dict_stand ()
    
```

Normalise un dictionnaire de TS

```

get_data ()
    
```

Getter du dataset modifié

```

get_data_from_captor (cpt)
    
```

Retourne toutes les séries temporelles liées à un nom de capteur

Parameters :

- **cpt** : **String** Nom du capteur désiré

Returns :

res : **{Dict}** Sous dataset du capteur associé, les clés sont les différentes déclinaisons en séries temporelles du capteur, varie en longueur selon la granularité

import_dataset ()
Appel a la factory pour recuperer les donnees

info ()
Permet d'avoir une idee des donnees du dataset en prenant une TS au hasard

normalize (*data*)
Normalisation des TS, moyenne : 0 et ecart type : 1 Data : dataframe

reset_dataset ()
Retourne aux donnees importees avant modifications

reset_days ()
Par default decoupe les TS dans la granularite maximale

reset_months ()
Par default decoupe les TS dans la granularite maximale

reset_setup ()
Full reset des variables de granularites

reset_years ()
Par default decoupe les TS dans la granularite maximale

smooth (*data*, *wind*, *col*)
Smooth via rolling window

Parameters :

- **data : DataFrame** La DF a smooth, attention a bien choisir la colonne voulu
- **wind : int** Taille de fenetre
- **col : String** La colonne de la DF a smooth

Returns :

data : DataFrame La DF remaniee

split_all ()

split_data_months ()
Decoupage des TS selon la variable de mois

split_data_weeks ()
Decoupage des TS selon les semaines

split_data_years ()
Decoupage des TS selon la variable d'annees

split_each_steps ()

split_year_month_multi_day (*dataset*, *month*)

split_year_multi_month ()

split_year_multi_month_multi_day ()

standardize (*data*)
Standardize des TS, moyenne : 0 et ecart type : 1 Data : dataframe

2.1.14 utils.statics_func module

`utils.statics_func.openfile_dialog` ()
Boite de dialogue permettant de recuperer les fichiers de sauvegarde de partitionnement Pickle

Parameters : NA

Returns :

fname : String Chemin vers le fichier selectionne

`utils.statics_func.thresholding_algo` (*y*, *lag*, *threshold*, *influence*)
Algorithm de detection de pics dans une TS

Parameters :

- lag
- threshold
- influence

Returns :

unnamed : {Dict} Informations sortie de l’algo, signals est une liste de meme len que la TS, representant les ppics via une variation boolean

2.1.15 Module contents

CHAPITRE 3

Installation

Téléchargement via Git :

```
git clone https://github.com/LevavasseurYann/workspace_internship.git
```

Vous trouverez les données réelles utilisées >>> <https://drive.google.com/open?id=1c0ZEve1EDBKlyPgjcAulyUEVAXNNWo4t>

Note : Ce projet est développé dans un environnement conda, un fichier spec-file est à télécharger ici >>> `txt`

Des questions ? Besoin d'aide ? Rejoignez le chat :

—

U

- utils, 16
- utils.cluster_ts, 5
- utils.data_factory, 8
- utils.geo, 9
- utils.k_mean, 10
- utils.k_shape, 10
- utils.parser, 11
- utils.plot, 11
- utils.prefix_span, 12
- utils.pygapbide, 13
- utils.sax_engine, 13
- utils.series_supp, 14
- utils.statics_func, 15

A

aff_color() (méthode utils.cluster_ts.ClusterTs), 6

B

backward_check() (méthode utils.pygapbide.Gapbide), 13

C

capteur_parser() (méthode utils.cluster_ts.ClusterTs), 6
 change_mode() (méthode utils.plot.Plot), 11
 check_equal() (méthode utils.cluster_ts.ClusterTs), 6
 clean_import_db() (méthode
 utils.pygapbide.GapbideManager), 13
 clust_hoverview() (méthode utils.cluster_ts.ClusterTs), 6
 clust_hoverview_rng() (méthode
 utils.cluster_ts.ClusterTs), 6
 cluster_counter() (méthode utils.k_mean.Kmean), 10
 cluster_counter() (méthode utils.k_shape.Kshape), 11
 ClusterTs (classe dans utils.cluster_ts), 5

D

DataFactory (classe dans utils.data_factory), 8
 dict_norm() (méthode utils.series_supp.SeriesSupp), 14
 dict_round() (méthode utils.series_supp.SeriesSupp), 14
 dict_smooth() (méthode utils.series_supp.SeriesSupp), 14
 dict_stand() (méthode utils.series_supp.SeriesSupp), 14
 distance_dict() (méthode utils.geo.Geo), 9
 distance_matrix() (méthode utils.geo.Geo), 9

E

export_format() (méthode
 utils.prefix_span.PrefixSpanManager), 12
 export_format() (méthode utils.sax_engine.SaxEngine), 13

F

fit() (méthode utils.sax_engine.SaxEngine), 13
 fit_run() (méthode utils.sax_engine.SaxEngine), 14
 forward_check() (méthode utils.pygapbide.Gapbide), 13

frequent() (méthode utils.prefix_span.PrefixSpanManager), 12

G

Gapbide (classe dans utils.pygapbide), 13
 GapbideManager (classe dans utils.pygapbide), 13
 gen_11_patterns() (méthode utils.pygapbide.Gapbide), 13
 Geo (classe dans utils.geo), 9
 get_captor_distribution_in_cluster() (méthode
 utils.cluster_ts.ClusterTs), 6
 get_clust_part_for_captor() (méthode
 utils.cluster_ts.ClusterTs), 6
 get_cluster_n() (méthode utils.cluster_ts.ClusterTs), 7
 get_data() (méthode utils.data_factory.DataFactory), 8
 get_data() (méthode utils.series_supp.SeriesSupp), 14
 get_data_from_captor() (méthode
 utils.series_supp.SeriesSupp), 14
 get_dataset() (méthode utils.data_factory.DataFactory), 8
 get_geo_data() (méthode utils.geo.Geo), 9
 get_GW() (méthode utils.data_factory.DataFactory), 8
 get_minrange_rg() (méthode utils.geo.Geo), 9
 get_part_of_ts() (méthode utils.cluster_ts.ClusterTs), 7
 get_RG1() (méthode utils.data_factory.DataFactory), 8
 get_RG24() (méthode utils.data_factory.DataFactory), 8
 get_ts_by_captor() (méthode utils.cluster_ts.ClusterTs), 7

H

highlight_max() (méthode utils.cluster_ts.ClusterTs), 7
 highlight_min() (méthode utils.geo.Geo), 9

I

import_dataset() (méthode utils.series_supp.SeriesSupp), 15
 info() (méthode utils.series_supp.SeriesSupp), 15

K

k_fit() (méthode utils.k_mean.Kmean), 10
 k_fit() (méthode utils.k_shape.Kshape), 11
 k_init() (méthode utils.k_mean.Kmean), 10

k_init() (méthode utils.k_shape.Kshape), 11
 Kmean (classe dans utils.k_mean), 10
 Kshape (classe dans utils.k_shape), 10

N

normalize() (méthode utils.series_supp.SeriesSupp), 15

O

openfile_dialog() (dans le module utils.statics_func), 15
 output() (méthode utils.pygapbide.Gapbide), 13

P

parse_capteur_split() (méthode
 utils.cluster_ts.ClusterTs), 7
 Parser (classe dans utils.parser), 11
 path_GW (attribut utils.data_factory.DataFactory), 9
 path_RG1 (attribut utils.data_factory.DataFactory), 9
 path_RG24 (attribut utils.data_factory.DataFactory), 9
 Plot (classe dans utils.plot), 11
 plot() (méthode utils.prefix_span.PrefixSpanManager), 12
 plot_2D() (méthode utils.geo.Geo), 9
 plot_3D() (méthode utils.geo.Geo), 9
 plot_3D_for_all_cluster() (méthode utils.geo.Geo), 10
 plot_3D_for_one_cluster() (méthode utils.geo.Geo), 10
 plot_captor_cluster_cover() (méthode utils.plot.Plot), 11
 plot_cluster() (méthode utils.plot.Plot), 12
 plot_cluster_light() (méthode utils.plot.Plot), 12
 plot_histo() (méthode utils.plot.Plot), 12
 plot_histo_2() (méthode utils.plot.Plot), 12
 plot_prefixspan() (méthode utils.plot.Plot), 12
 plot_scatter() (méthode utils.plot.Plot), 12
 plot_scatter_by_capteur() (méthode utils.plot.Plot), 12
 plot_scatter_by_capteur_color_cluster() (méthode
 utils.plot.Plot), 12
 plot_scatter_light() (méthode utils.plot.Plot), 12
 plot_simple_list() (méthode utils.plot.Plot), 12
 plot_simple_TS() (méthode utils.plot.Plot), 12
 plotly_3D() (méthode utils.geo.Geo), 10
 PrefixSpanManager (classe dans utils.prefix_span), 12

R

read_cluster() (méthode utils.cluster_ts.ClusterTs), 7
 reset() (méthode utils.sax_engine.SaxEngine), 14
 reset_dataset() (méthode utils.series_supp.SeriesSupp),
 15
 reset_days() (méthode utils.series_supp.SeriesSupp), 15
 reset_gp() (méthode utils.pygapbide.GapbideManager),
 13
 reset_months() (méthode utils.series_supp.SeriesSupp),
 15
 reset_setup() (méthode utils.series_supp.SeriesSupp), 15
 reset_years() (méthode utils.series_supp.SeriesSupp), 15
 run() (méthode utils.prefix_span.PrefixSpanManager), 12

run() (méthode utils.pygapbide.Gapbide), 13
 run() (méthode utils.pygapbide.GapbideManager), 13
 run() (méthode utils.sax_engine.SaxEngine), 14

S

sax_to_spmf() (méthode utils.parser.Parser), 11
 SaxEngine (classe dans utils.sax_engine), 13
 SeriesSupp (classe dans utils.series_supp), 14
 set_nb_segment() (méthode utils.sax_engine.SaxEngine),
 14
 set_nb_symbol() (méthode utils.sax_engine.SaxEngine),
 14
 set_size_min() (méthode utils.cluster_ts.ClusterTs), 7
 show_info() (méthode utils.cluster_ts.ClusterTs), 7
 smooth() (méthode utils.series_supp.SeriesSupp), 15
 span() (méthode utils.pygapbide.Gapbide), 13
 split_all() (méthode utils.series_supp.SeriesSupp), 15
 split_data_months() (méthode
 utils.series_supp.SeriesSupp), 15
 split_data_weeks() (méthode
 utils.series_supp.SeriesSupp), 15
 split_data_years() (méthode
 utils.series_supp.SeriesSupp), 15
 split_each_steps() (méthode
 utils.series_supp.SeriesSupp), 15
 split_year_month_multi_day() (méthode
 utils.series_supp.SeriesSupp), 15
 split_year_multi_month() (méthode
 utils.series_supp.SeriesSupp), 15
 split_year_multi_month_multi_day() (méthode
 utils.series_supp.SeriesSupp), 15
 standardize() (méthode utils.series_supp.SeriesSupp), 15
 step_run() (méthode utils.sax_engine.SaxEngine), 14
 store_cluster() (méthode utils.cluster_ts.ClusterTs), 8
 style_df() (méthode utils.cluster_ts.ClusterTs), 8

T

thresholding_algo() (dans le module utils.statics_func),
 15
 topk() (méthode utils.prefix_span.PrefixSpanManager),
 12
 tslearn_format_export() (méthode
 utils.cluster_ts.ClusterTs), 8

U

utils (module), 16
 utils.cluster_ts (module), 5
 utils.data_factory (module), 8
 utils.geo (module), 9
 utils.k_mean (module), 10
 utils.k_shape (module), 10
 utils.parser (module), 11
 utils.plot (module), 11
 utils.prefix_span (module), 12

utils.pygapbide (module), 13
utils.sax_engine (module), 13
utils.series_supp (module), 14
utils.statics_func (module), 15