# Module Smart-house-REST-API.Clustering

## Sub-modules

* Smart-house-REST-API.Clustering.K\_MEAN
* Smart-house-REST-API.Clustering.controller
* Smart-house-REST-API.Clustering.losses
* Smart-house-REST-API.Clustering.tests
* Smart-house-REST-API.Clustering.utils Module Smart-house-REST-API.Clustering.controller ================================================= Module Smart-house-REST-API.Clustering.K\_MEAN =============================================

## Classes

Model(X\_train, n\_clusters=3, loss='distance')

Init function of obcject K\_Means.Model

Args: X\_train (iterable): Iterable object that represents data to be clustered n\_clusters (int, optional): [description]. Defaults to 3. loss (str, optional): [description]. Defaults to ‘distance’.

### Methods

fit(self, epochs=50, verbose=2)

Method for fitting K-Mean algorithm to the dataset

Args: epochs (int, optional): Max number of epochs. Defaults to 50. verbose (int, optional): Level of printing –> higher level means more information. Defaults to 2.

predict(self, X)

Method that predicts to witch cluster belong each element of provided X

Args: X (iterable): data to be classified into clusters

Returns: iterable: array of predicted clusters Module Smart-house-REST-API.Clustering.losses =============================================

## Functions

geometric\_distance(a, b, dist=0)

Loss function calculating geometric distance between 2 objects. It can be explained by the example of distance between two points on 2D plane.

^ | .B(x2, y2) | / | / | .A(x1, y1) | + ——————— >

It is derived by: sqrt((x1 - x2)^2 + (y1 - y2)^2) This implementation of distance metric is multidimensional, which enables calculation of geometric distance between

Args: a (object): first object b (object): second object of same length as object a dist (int, optional): [description]. Defaults to 0.

Returns: float: value of geometric distance Module Smart-house-REST-API.Clustering.utils ============================================

## Functions

is\_iter(variable)

Function that can be used to find wheter object is iterable

Args: variable (object): ariable to be checked

Returns: True: variable is iterable False: variable is not iterable

from losses import is\_iter is\_iter([1,2]) True is\_iter(“hello”) is\_iter(6) False

recursive\_list(array)

Function creating a list from iterable object of any shape

Args: array (iterable): iterable object to be changed into list

Returns: list: array of type list

from utils import recursive\_list tab = [[[1, 2], 3] , [[-1, 5], 6], [[4, 2], -9]] recursive\_list(tab) [[[1, 2], 3], [[-1, 5], 6], [[4, 2], -9]]

recursive\_max(array, max\_val=None)

Function that returns maximum value of an interable object of any shape

Args: array (iterable): array/list/iterable object of any shape max\_val ([type], optional): in-function parameter. Defaults to None. Do not change.

Returns: float: maximum found in the array

from utils import recursive\_max tab = [[[1, 2], 3] , [[-1, 5], 6], [[4, 2], -9]] recursive\_max(tab) 6

recursive\_mean(array)

Function calculationg mean point from all objects in an iterable array.

Args: array (iterable): array of objects of the same shape

Returns: object: mean object of the array

from utils import recursive\_mean tab = [[[1, 2], 3] , [[-1, 5], 6], [[4, 2], -9]] recursive\_mean(tab) [[1.3333333333333333, 3.0], 0.0]

recursive\_min(array, min\_val=None)

Function that returns minimum value of an interable object of any shape

Args: array (iterable): array/list/iterable object of any shape min\_val ([type], optional): in-function parameter. Defaults to None. Do not change.

Returns: float: minimum found in the array

from utils import recursive\_max tab = [[[1, 2], 3] , [[-1, 5], 6], [[4, 2], -9]] recursive\_max(tab) -9

recursive\_nparray(array)

Function creating a numpy array from iterable object of any shape

Args: array (iterable): iterable object to be changed into numpy array

Returns: array: array of type numpy\_array

from utils import recursive\_nparray tab = [[[1, 2], 3] , [[-1, 5], 6], [[4, 2], -9]] recursive\_nparray(tab) array([[array([1, 2]), 3], [array([-1, 5]), 6], [array([4, 2]), -9]], dtype=object)