# Clustering Group T4-2

1.0

Generated by Doxygen 1.9.1

**1 Description** 1

**2 Hierarchical Index** 3

2.1 Class Hierarchy . . . 3

**3 Class Index** 5

3.1 Class List . . . 5

**4 Class Documentation** 7

4.1 clustering.Clustering Class Reference . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Member Function Documentation . . . 8

4.1.2.1 cluster() . . . 8

4.1.2.2 house_load() . . . 8

4.1.2.3 pyc_metric() . . . 8

4.2 dbscan.DBSCANClustering Class Reference . . . 9

4.2.1 Detailed Description . . . 9

4.2.2 Member Function Documentation . . . 9

4.2.2.1 cluster() . . . 9

4.2.2.2 package() . . . 10

4.3 indices.Indices Class Reference . . . 10

4.4 kmeans.kmeansClustering Class Reference . . . 11

4.4.1 Detailed Description . . . 11

4.4.2 Member Function Documentation . . . 11

4.4.2.1 cluster() . . . 11

4.5 kmedians.kmediansClustering Class Reference . . . 12

4.5.1 Detailed Description . . . 12

4.5.2 Member Function Documentation . . . 12

4.5.2.1 cluster() . . . 12

4.6 kmedoids.kmedoidsClustering Class Reference . . . 13

4.6.1 Detailed Description . . . 13

4.6.2 Member Function Documentation . . . 14

4.6.2.1 cluster() . . . 14

4.6.2.2 package() . . . 14

**Index** 15

# Chapter 1

# Description

Group Project T4-2 done for the Lecture Data Science 1 at the Goethe University Frankfurt


- Niklas Conen
- Jonas Elpelt
- Franziska Hicking
- Julian Rummel

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:


clustering.Clustering . . . . . 7
dbscan.DBSCANClustering . . . . . 9
kmeans.kmeansClustering . . . . . 11
kmedians.kmediansClustering . . . . . 12
kmedoids.kmedoidsClustering . . . . . 13
indices.Indices . . . . . 10

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


clustering.Clustering
: Meta Class for all subsequent clustering algorithms implements all functions needed for running the different cluster algorithms . . . 7

dbscan.DBSCANClustering
: Implements DBSCAN Clustering uses the scikit-learn DBSCAN implementation . . . 9

indices.Indices . . . 10

kmeans.kmeansClustering
: Class implementing k-Means Clustering uses the pyclustering k-means implementation centers can be initialised using the k++ or the random initialiser . . . 11

kmedians.kmediansClustering
: Implements k-Medians Clustering uses the pyclustering k-medians implementation centers are initialised using the random initialiser . . . 12

kmedoids.kmedoidsClustering
: Implements k-Medians Clustering uses the scikit-learn-extra k-medoids implementation centers are set using the k++ initialiser if not set differently . . . 13

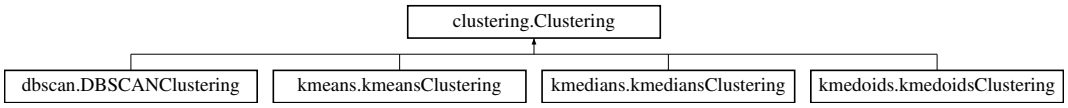
# Chapter 4

# Class Documentation

## 4.1 clustering.Clustering Class Reference

Meta Class for all subsequent clustering algorithms implements all functions needed for running the different cluster algorithms.

Inheritance diagram for clustering.Clustering:




### Public Member Functions

- def **__init__** (self, metric, dataset)
- def pyc_metric (self, metric)

  *returns a distance metric which is usable by the pyclustering algorithms*
- def load_data (self)

  *loads in a dataset, standardises it and sets it as self.data attribute*
- def house_load (self, path, skip=1)

  *loads the housevotes dataset and encodes it using One-Hot-Encoding*
- def cluster (self)

  *does nothing in the meta class.*

### Public Attributes

- **data**
- **dataset**
- **labels**

### 4.1.1 Detailed Description

Meta Class for all subsequent clustering algorithms implements all functions needed for running the different cluster algorithms.

### 4.1.2 Member Function Documentation

#### 4.1.2.1 cluster()

```
def clustering.Clustering.cluster (
              self )
```

does nothing in the meta class.

needs to be implemented in the inheriting cluster algorithm classes

#### 4.1.2.2 house_load()

```
def clustering.Clustering.house_load (
              self,
              path,
              skip = 1 )
```

loads the housevotes dataset and encodes it using One-Hot-Encoding

**Parameters**

| | |
|---|---|
| *path* | filepath to the dataset |
| *skip* | number of lines that get skipped when reading in a file |

**Returns**

One-Hot-Encoded housevotes dataset

#### 4.1.2.3 pyc_metric()

```
def clustering.Clustering.pyc_metric (
              self,
              metric )
```

returns a distance metric which is usable by the pyclustering algorithms

**Parameters**

| | |
|---|---|
| *distance* | metric string. allowed: "euclidean", "manhattan", "chebyshev", "cosine" |

**Returns**

pyclustering distance_metric object, None when distance is not supported

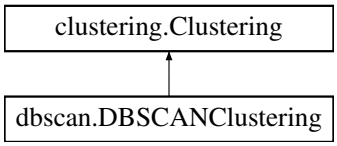
The documentation for this class was generated from the following file:

- clustering.py

## 4.2 dbscan.DBSCANClustering Class Reference

implements DBSCAN Clustering uses the scikit-learn DBSCAN implementation

Inheritance diagram for dbscan.DBSCANClustering:




### Public Member Functions

- def **__init__** (self, metric, dataset)
- def cluster (self, eps, minPts)

  *clustering method.*
- def package (self, labels)

  *rearranges the result to a format similar to the one of the pyclustering algorithms allows for easier access in the streamlit interface*

### Public Attributes

- **metric**

### 4.2.1 Detailed Description

implements DBSCAN Clustering uses the scikit-learn DBSCAN implementation

### 4.2.2 Member Function Documentation

#### 4.2.2.1 cluster()

```
def dbscan.DBSCANClustering.cluster (
              self,
              eps,
              minPts )
```

clustering method.

Will execute clustering on the data saved in self.data with the metric given in self.metric params are the same as in the DBSCAN paper

**Parameters**

| | |
|---|---|
| *eps* | Distance for the Eps-Neighbourhood |
| *minPts* | Minmal number of points in a cluster |

**Returns**

formatted clustered data

#### 4.2.2.2 package()

```
def dbscan.DBSCANClustering.package (
              self,
              labels )
```

rearranges the result to a format similar to the one of the pyclustering algorithms allows for easier access in the streamlit interface

**Parameters**

| | |
|---|---|
| *labels* | cluster labels DBSCAN assigns to a point |

**Returns**

clusters as list of lists of indices of points and noise as list of indices of points

The documentation for this class was generated from the following file:

- dbscan.py

## 4.3 indices.Indices Class Reference

### Public Member Functions

- def **__init__** (self, cluster_calc, cluster_label)
- def **index_external** (self, index)
- def **index_internal** (self, index)

### Public Attributes

- **cluster_calc**
- **cluster_label**

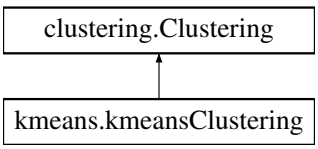
The documentation for this class was generated from the following file:

- indices.py

## 4.4 kmeans.kmeansClustering Class Reference

Class implementing k-Means Clustering uses the pyclustering k-means implementation centers can be initialised using the k++ or the random initialiser.

Inheritance diagram for kmeans.kmeansClustering:




### Public Member Functions

- def **__init__** (self, metric, dataset)
- def cluster (self, k, plusplus=True)

  *clustering method.*

### Public Attributes

- **data**
- **metric**

### 4.4.1 Detailed Description

Class implementing k-Means Clustering uses the pyclustering k-means implementation centers can be initialised using the k++ or the random initialiser.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 cluster()

```
def kmeans.kmeansClustering.cluster (
              self,
              k,
              plusplus = True )
```

clustering method.

Will execute clustering on the data saved in self.data with the metric given in self.metric

**Parameters**

| | |
|---|---|
| *k* | number of clusters that are generated |
| *plusplus* | will use k++ initialiser if true |

**Returns**

clusters as list of lists of indices of points and final cluster centers

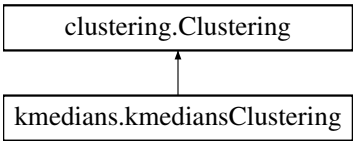
The documentation for this class was generated from the following file:

- kmeans.py

## 4.5 kmedians.kmediansClustering Class Reference

implements k-Medians Clustering uses the pyclustering k-medians implementation centers are initialised using the random initialiser

Inheritance diagram for kmedians.kmediansClustering:




### Public Member Functions

- def **__init__** (self, metric, dataset)
- def cluster (self, k)

  *clustering method.*

### Public Attributes

- **data**
- **metric**

### 4.5.1 Detailed Description

implements k-Medians Clustering uses the pyclustering k-medians implementation centers are initialised using the random initialiser

### 4.5.2 Member Function Documentation

#### 4.5.2.1 cluster()

```
def kmedians.kmediansClustering.cluster (
              self,
              k )
```

clustering method.

Will execute clustering on the data saved in self.data with the metric given in self.metric

**Parameters**

| | |
|---|---|
| *k* | number of clusters that are generated |

**Returns**

clusters as list of lists of indices of points and final cluster medians

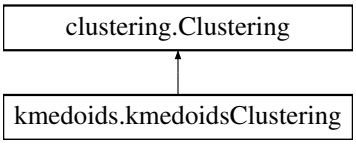
The documentation for this class was generated from the following file:

- kmedians.py

## 4.6 kmedoids.kmedoidsClustering Class Reference

implements k-Medians Clustering uses the scikit-learn-extra k-medoids implementation centers are set using the k++ initialiser if not set differently

Inheritance diagram for kmedoids.kmedoidsClustering:




### Public Member Functions

- def **__init__** (self, metric, dataset)
- def cluster (self, k, init="k-medoids++")

  *clustering method.*
- def package (self, labels)

  *rearranges the result to a format similar to the one of the pyclustering algorithms allows for easier access in the streamlit interface*

### Public Attributes

- **data**
- **metric**

### 4.6.1 Detailed Description

implements k-Medians Clustering uses the scikit-learn-extra k-medoids implementation centers are set using the k++ initialiser if not set differently

### 4.6.2 Member Function Documentation

#### 4.6.2.1 cluster()

```
def kmedoids.kmedoidsClustering.cluster (
              self,
              k,
              init = "k-medoids++" )
```

clustering method.

Will execute clustering on the data saved in self.data with the metric given in self.metric

**Parameters**

| | |
|---|---|
| *k* | number of clusters that are generated |
| *init* | initialisation parameter. Standard: "k-medoids++" |

**Returns**

clusters as list of lists of indices of points, final cluster centers

#### 4.6.2.2 package()

```
def kmedoids.kmedoidsClustering.package (
              self,
              labels )
```

rearranges the result to a format similar to the one of the pyclustering algorithms allows for easier access in the streamlit interface

**Parameters**

| | |
|---|---|
| *labels* | labels returned from the KMedoids algorithm |

**Returns**

clusters formated similarly to the pyclustering algorithms

The documentation for this class was generated from the following file:

- kmedoids.py

# Index

cluster
    clustering.Clustering, 8
    dbscan.DBSCANClustering, 9
    kmeans.kmeansClustering, 11
    kmedians.kmediansClustering, 12
    kmedoids.kmedoidsClustering, 14
clustering.Clustering, 7
    cluster, 8
    house_load, 8
    pyc_metric, 8

dbscan.DBSCANClustering, 9
    cluster, 9
    package, 10

house_load
    clustering.Clustering, 8

indices.Indices, 10

kmeans.kmeansClustering, 11
    cluster, 11
kmedians.kmediansClustering, 12
    cluster, 12
kmedoids.kmedoidsClustering, 13
    cluster, 14
    package, 14

package
    dbscan.DBSCANClustering, 10
    kmedoids.kmedoidsClustering, 14
pyc_metric
    clustering.Clustering, 8