# Distance Measures and Clustering Group T4-2

1.0

Generated by Doxygen 1.9.1

# 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

So long, and thanks for all the fish

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

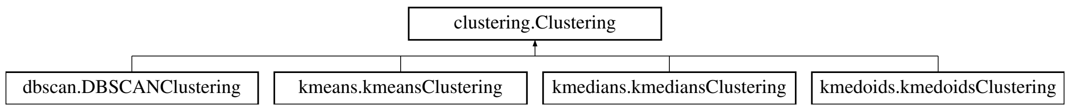
# Chapter 4

# 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)

    *constructor*
- 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*
    *democrats are labeled as 1, republicans as 0*
- def cluster (self)

    *does nothing in the meta class.*

### Public Attributes

- metric

    *metric name as string or pyclustering distance_metric object*
- dataset

    *dataset name as string*
- data

    *data that gets clustered*
- labels

    *expected cluster values*

### 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 Constructor & Destructor Documentation

#### 4.1.2.1 __init__()

```
def clustering.Clustering.__init__ (
              self,
              metric,
              dataset )
```

constructor

**Parameters**

| | |
|---|---|
| *metric* | metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine" |
| *dataset* | dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes" |

Reimplemented in kmedoids.kmedoidsClustering, kmedians.kmediansClustering, kmeans.kmeansClustering, and dbscan.DBSCANClustering.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 cluster()

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

does nothing in the meta class.

needs to be implemented in the inheriting cluster algorithm classes

#### 4.1.3.2 house_load()

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

loads the housevotes dataset and encodes it using One-Hot-Encoding
democrats are labeled as 1, republicans as 0

**Parameters**

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

**Returns**

One-Hot-Encoded housevotes dataset and labels as array of 1s and 0s

#### 4.1.3.3 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

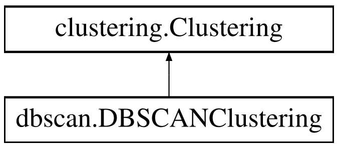
The documentation for this class was generated from the following file:

- 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)

  *constructor*
- 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

  *metric name as string*
- dataset

  *dataset name as string*
- data

  *data that gets clustered*
- labels

  *expected cluster values*

### 4.2.1 Detailed Description

implements DBSCAN Clustering
uses the scikit-learn DBSCAN implementation

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 __init__()

```
def dbscan.DBSCANClustering.__init__ (
              self,
              metric,
              dataset )
```

constructor

**Parameters**

| | |
|---|---|
| *metric* | metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine" |
| *dataset* | dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes" |

Reimplemented from clustering.Clustering.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 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

#### 4.2.3.2 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**

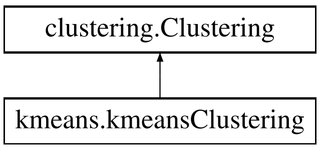
The documentation for this class was generated from the following file:

- 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)

  *constructor*
- def cluster (self, k, plusplus=True)

  *clustering method.*

### Public Attributes

- metric

  *metric name as pyclustering distance_metric object*
- dataset

  *dataset name as string*
- data

  *data that gets clustered*
- labels

  *expected cluster values*

### 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 Constructor & Destructor Documentation

#### 4.4.2.1 __init__()

```
def kmeans.kmeansClustering.__init__ (
              self,
              metric,
              dataset )
```

constructor

**Parameters**

| | |
|---|---|
| *metric* | metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine" |
| *dataset* | dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes" |

Reimplemented from clustering.Clustering.

### 4.4.3 Member Function Documentation

#### 4.4.3.1 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

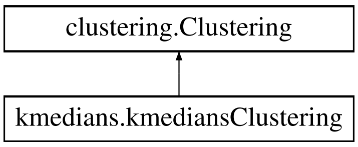
The documentation for this class was generated from the following file:

- 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)

  *constructor*
- def cluster (self, k)

  *clustering method.*

### Public Attributes

- metric

  *metric name as pyclustering distance_metric object*
- dataset

  *dataset name as string*
- data

  *data that gets clustered*
- labels

  *expected cluster values*

### 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 Constructor & Destructor Documentation

#### 4.5.2.1 __init__()

```
def kmedians.kmediansClustering.__init__ (
              self,
              metric,
              dataset )
```

constructor

**Parameters**

| | |
|---|---|
| *metric* | metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine" |
| *dataset* | dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes" |

Reimplemented from clustering.Clustering.

### 4.5.3 Member Function Documentation

#### 4.5.3.1 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

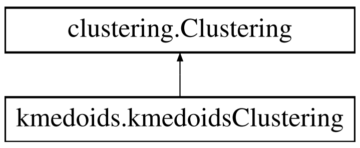
The documentation for this class was generated from the following file:

- 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)

  *constructor*
- 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

- metric

  *metric name as string*
- dataset

  *dataset name as string*
- data

  *data that gets clustered*
- labels

  *expected cluster values*

## 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 Constructor & Destructor Documentation

#### 4.6.2.1 __init__()

```
def kmedoids.kmedoidsClustering.__init__ (
              self,
              metric,
              dataset )
```

constructor

**Parameters**

| | |
|---|---|
| *metric* | metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine" |
| *dataset* | dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes" |

Reimplemented from clustering.Clustering.

### 4.6.3 Member Function Documentation

#### 4.6.3.1 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. Default: "k-medoids++" |

**Returns**

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

#### 4.6.3.2 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