

Distance Measures and Clustering Group T4-2

1.0

Generated by Doxygen 1.9.1

1 Description	1
1.1 Frontend	1
1.2 Documentation	1
1.3 Dependencies and Sources	1
1.4 Todos	2
1.5 Nice-To-Haves	2
1.6 Authors	2
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 clustering Namespace Reference	11
6.2 comparison_plots Namespace Reference	11
6.2.1 Variable Documentation	12
6.2.1.1 all_dist	12
6.2.1.2 all_k	12
6.2.1.3 all_kalgos	12
6.2.1.4 all_kalgos_df	13
6.2.1.5 all_kalgos_int	13
6.2.1.6 ax	13
6.2.1.7 bbox_to_anchor	13
6.2.1.8 borderaxespad	13
6.2.1.9 c	13
6.2.1.10 cluster	13
6.2.1.11 clustered_data	14
6.2.1.12 clusters	14
6.2.1.13 d	14
6.2.1.14 data	14
6.2.1.15 datasets	14
6.2.1.16 distances	14
6.2.1.17 fig	14
6.2.1.18 figsize	14
6.2.1.19 hue	15
6.2.1.20 l1	15

6.2.1.21 index	15
6.2.1.22 index_ext_eval	15
6.2.1.23 index_int_eval	15
6.2.1.24 index_score	15
6.2.1.25 index_scores	15
6.2.1.26 k	16
6.2.1.27 kalgoclass	16
6.2.1.28 kalgos	16
6.2.1.29 kind	16
6.2.1.30 labels	16
6.2.1.31 legend	16
6.2.1.32 loc	16
6.2.1.33 num_of_classes	17
6.2.1.34 predicted	17
6.2.1.35 results	17
6.2.1.36 s	17
6.2.1.37 seed	17
6.2.1.38 stuff	17
6.2.1.39 style	17
6.2.1.40 title	17
6.2.1.41 x	18
6.2.1.42 y	18
6.3 dbscan Namespace Reference	18
6.4 dbscan_heuristic Namespace Reference	18
6.5 heuristic_web Namespace Reference	18
6.5.1 Variable Documentation	19
6.5.1.1 cluster_dist	19
6.5.1.2 cluster_dist_desc	19
6.5.1.3 col1	19
6.5.1.4 col2	19
6.5.1.5 dataset	19
6.5.1.6 df	19
6.5.1.7 heu	20
6.5.1.8 k	20
6.5.1.9 kdist	20
6.5.1.10 key	20
6.5.1.11 line	20
6.5.1.12 nearest	20
6.5.1.13 page_icon	20
6.5.1.14 page_title	21
6.5.1.15 points	21
6.5.1.16 reverse	21

6.5.1.17 rules	21
6.5.1.18 selectors	21
6.5.1.19 submit_button	21
6.5.1.20 text	21
6.5.1.21 textp	22
6.5.1.22 use_container_width	22
6.5.1.23 yaxis	22
6.6 indices Namespace Reference	22
6.7 kmeans Namespace Reference	22
6.8 kmedians Namespace Reference	22
6.9 kmedoids Namespace Reference	23
6.10 result_calculation Namespace Reference	23
6.10.1 Variable Documentation	23
6.10.1.1 alg	23
6.10.1.2 c	23
6.10.1.3 centers	24
6.10.1.4 clusters	24
6.10.1.5 d	24
6.10.1.6 datasets	24
6.10.1.7 distances	24
6.10.1.8 eps	24
6.10.1.9 k	24
6.10.1.10 kalgoclass	24
6.10.1.11 kalgos	25
6.10.1.12 m	25
6.10.1.13 minpts	25
6.10.1.14 results	25
6.10.1.15 s	25
6.10.1.16 seed	25
6.11 results Namespace Reference	25
6.12 SessionState Namespace Reference	26
6.12.1 Function Documentation	26
6.12.1.1 get()	26
6.13 web_frontend Namespace Reference	27
6.13.1 Variable Documentation	28
6.13.1.1 add_result	28
6.13.1.2 align	28
6.13.1.3 alpha	29
6.13.1.4 altcolor	29
6.13.1.5 angles	29
6.13.1.6 ax	29
6.13.1.7 cluster	29

6.13.1.8 cluster_algo	29
6.13.1.9 cluster_algo_class	29
6.13.1.10 cluster_dist	30
6.13.1.11 cluster_dist_desc	30
6.13.1.12 cluster_label	30
6.13.1.13 clustered_data	30
6.13.1.14 clusters	30
6.13.1.15 col1	30
6.13.1.16 col2	30
6.13.1.17 color	31
6.13.1.18 color_palette	31
6.13.1.19 dataexpander	31
6.13.1.20 dataset	31
6.13.1.21 datasets	31
6.13.1.22 desc	31
6.13.1.23 desc_list	31
6.13.1.24 df	32
6.13.1.25 dfclusterdata	32
6.13.1.26 eps	32
6.13.1.27 epsilon	32
6.13.1.28 False	32
6.13.1.29 fig	32
6.13.1.30 hue	32
6.13.1.31 I1	33
6.13.1.32 index_eval	33
6.13.1.33 indices_data	33
6.13.1.34 k	33
6.13.1.35 k_value	33
6.13.1.36 labels	33
6.13.1.37 legend	33
6.13.1.38 linewidth	34
6.13.1.39 markers	34
6.13.1.40 marking_centroids	34
6.13.1.41 minpts	34
6.13.1.42 page_icon	34
6.13.1.43 page_title	34
6.13.1.44 palette	34
6.13.1.45 pcaalt	35
6.13.1.46 perp	35
6.13.1.47 point_label	35
6.13.1.48 precalc	35
6.13.1.49 predicted	35

6.13.1.50 projected_data_pca	35
6.13.1.51 projected_data_tsne	35
6.13.1.52 reset_tmp	36
6.13.1.53 resulthandler	36
6.13.1.54 results	36
6.13.1.55 score	36
6.13.1.56 seaplots	36
6.13.1.57 seed	36
6.13.1.58 seeded	36
6.13.1.59 session_state	37
6.13.1.60 size	37
6.13.1.61 stats	37
6.13.1.62 stuff	37
6.13.1.63 style	37
6.13.1.64 subplot_kw	37
6.13.1.65 tsnealt	37
6.13.1.66 use_container_width	38
6.13.1.67 val	38
6.13.1.68 weights	38
6.13.1.69 width	38
6.13.1.70 x	38
6.13.1.71 xaxis	38
6.13.1.72 xs	38
6.13.1.73 y	38
6.13.1.74 yaxis	38
6.13.1.75 ys	38
7 Class Documentation	39
7.1 clustering.Clustering Class Reference	39
7.1.1 Detailed Description	40
7.1.2 Constructor & Destructor Documentation	40
7.1.2.1 __init__()	40
7.1.3 Member Function Documentation	40
7.1.3.1 cluster()	41
7.1.3.2 house_load()	41
7.1.3.3 load_data()	41
7.1.3.4 pyc_metric()	41
7.1.4 Member Data Documentation	42
7.1.4.1 data	42
7.1.4.2 datadf	42
7.1.4.3 dataset	42
7.1.4.4 labels	42

7.1.4.5 metric	43
7.1.4.6 seed	43
7.2 dbscan.DBSCANClustering Class Reference	43
7.2.1 Detailed Description	44
7.2.2 Constructor & Destructor Documentation	44
7.2.2.1 __init__()	44
7.2.3 Member Function Documentation	44
7.2.3.1 cluster()	44
7.2.3.2 package()	45
7.2.4 Member Data Documentation	45
7.2.4.1 data	45
7.2.4.2 dataset	45
7.2.4.3 labels	45
7.2.4.4 metric	46
7.3 dbscan_heuristic.DBSCANHeuristic Class Reference	46
7.3.1 Detailed Description	46
7.3.2 Constructor & Destructor Documentation	46
7.3.2.1 __init__()	47
7.3.3 Member Function Documentation	47
7.3.3.1 kdist()	47
7.3.3.2 plot_kdist()	47
7.3.3.3 set_dataset()	47
7.3.3.4 set_metric()	48
7.3.4 Member Data Documentation	48
7.3.4.1 clustering	48
7.3.4.2 k	48
7.3.4.3 metric	48
7.4 indices.Indices Class Reference	49
7.4.1 Detailed Description	49
7.4.2 Constructor & Destructor Documentation	49
7.4.2.1 __init__()	49
7.4.3 Member Function Documentation	50
7.4.3.1 index_external()	50
7.4.3.2 index_internal()	50
7.4.4 Member Data Documentation	50
7.4.4.1 cluster_calc	50
7.4.4.2 cluster_label	51
7.5 kmeans.kmeansClustering Class Reference	51
7.5.1 Detailed Description	51
7.5.2 Constructor & Destructor Documentation	52
7.5.2.1 __init__()	52
7.5.3 Member Function Documentation	52

7.5.3.1 cluster()	52
7.5.4 Member Data Documentation	52
7.5.4.1 data	53
7.5.4.2 dataset	53
7.5.4.3 labels	53
7.5.4.4 metric	53
7.5.4.5 seed	53
7.6 kmedians.kmediansClustering Class Reference	54
7.6.1 Detailed Description	54
7.6.2 Constructor & Destructor Documentation	54
7.6.2.1 __init__()	54
7.6.3 Member Function Documentation	55
7.6.3.1 cluster()	55
7.6.4 Member Data Documentation	55
7.6.4.1 data	55
7.6.4.2 dataset	55
7.6.4.3 labels	56
7.6.4.4 metric	56
7.6.4.5 seed	56
7.7 kmedoids.kmedoidsClustering Class Reference	56
7.7.1 Detailed Description	57
7.7.2 Constructor & Destructor Documentation	57
7.7.2.1 __init__()	57
7.7.3 Member Function Documentation	57
7.7.3.1 cluster()	58
7.7.3.2 package()	58
7.7.4 Member Data Documentation	58
7.7.4.1 data	58
7.7.4.2 dataset	59
7.7.4.3 labels	59
7.7.4.4 metric	59
7.7.4.5 seed	59
7.8 results.Results Class Reference	59
7.8.1 Detailed Description	60
7.8.2 Constructor & Destructor Documentation	60
7.8.2.1 __init__()	60
7.8.3 Member Function Documentation	60
7.8.3.1 get_path()	61
7.8.3.2 load_set()	61
7.8.3.3 save_set()	62
7.8.3.4 set_exists()	62
7.8.4 Member Data Documentation	62

7.8.4.1 parent	63
7.9 SessionState.SessionState Class Reference	63
7.9.1 Constructor & Destructor Documentation	63
7.9.1.1 __init__()	63
8 File Documentation	65
8.1 clustering.py File Reference	65
8.1.1 Detailed Description	65
8.2 comparison_plots.py File Reference	65
8.3 dbscan.py File Reference	66
8.3.1 Detailed Description	67
8.4 dbscan_heuristic.py File Reference	67
8.4.1 Detailed Description	67
8.5 heuristic_web.py File Reference	67
8.5.1 Detailed Description	68
8.6 indices.py File Reference	68
8.6.1 Detailed Description	69
8.7 kmeans.py File Reference	69
8.7.1 Detailed Description	69
8.8 kmedians.py File Reference	69
8.8.1 Detailed Description	70
8.9 kmedoids.py File Reference	70
8.9.1 Detailed Description	70
8.10 result_calculation.py File Reference	70
8.11 results.py File Reference	71
8.11.1 Detailed Description	71
8.12 SessionState.py File Reference	71
8.12.1 Detailed Description	71
8.12.1.1 Usage	72
8.13 web_frontend.py File Reference	72
8.13.1 Detailed Description	74
8.14 /home/nordegraf/Uni/8.Semester/DataScienceI/datascience1_group42/README.md File Reference	74
Index	75

Chapter 1

Description

This is a group project done for the Lecture "Data Science 1" at the Goethe University Frankfurt exploring the effects of different distance measures on distance-based clustering algorithm.

1.1 Frontend

The web frontend is accessible [here](#).

1.2 Documentation

The doxygen Documentation of the codebase can be accessed [here](#).

A PDF Documentation is also available in the docs directory (file: [documentation.pdf](#))

1.3 Dependencies and Sources

The [SessionState.py](#) is directly taken from a [gist](#) by Thiago Teixeira, user [tvst](#) on github. We take absolutely no credit for it!

The code for the altair chart used for displaying the DBSCAN heuristic is based heavily upon the multiline tooltip example from the altair example gallery ([Link](#)).

The project depends on following python libraries:

- [matplotlib](#)
- [numpy](#)
- [pandas](#)
- [pyclustering](#)
- [scikit-learn](#)
- [scikit-learn-extra](#)
- [seaborn](#)
- [streamlit](#)
- [altair](#)

1.4 ToDos

- [] Web Frontend Anleitung
- [] Web Frontend Doxygen Doku?
- [] Conclusion
- [] Abstract

1.5 Nice-To-Haves

- [] Dendrogramm
- [] Diskriminanzanalyse (Trennfähigste Variablen)

1.6 Authors

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

So long, and thanks for all the fish

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

clustering	11
comparison_plots	11
dbscan	18
dbscan_heuristic	18
heuristic_web	18
indices	22
kmeans	22
kmedians	22
kmedoids	23
result_calculation	23
results	25
SessionState	26
web_frontend	27

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

clustering.Clustering	39
dbscan.DBSCANClustering	43
kmeans.kmeansClustering	51
kmedians.kmediansClustering	54
kmedoids.kmedoidsClustering	56
dbscan_heuristic.DBSCANHeuristic	46
indices.Indices	49
object	
SessionState.SessionState	63
results.Results	59

Chapter 4

Class Index

4.1 Class List

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

clustering.Clustering	Base Class for all subsequent clustering algorithms implements all functions needed for running the different cluster algorithms	39
dbscan.DBSCANClustering	Implements DBSCAN Clustering uses the scikit-learn DBSCAN implementation	43
dbscan_heuristic.DBSCANHeuristic	Implements the DBSCAN heuristic proposed in the original DBSCAN paper:	46
indices.Indices	Calculates Indices for computed cluster labels uses the scikit library	49
kmeans.kmeansClustering	Class implementing k-Means Clustering uses the pyclustering k-means implementation centers can be initialised using the k++ or the random initialiser	51
kmedians.kmediansClustering	Implements k-Medians Clustering uses the pyclustering k-medians implementation centers are initialised using the random initialiser	54
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	56
results.Results	Class for easily saving and loading already calculated clustering results every dataset has a folder containing subfolders for every clustering algorithm containing more subfolders for every distance measure	59
SessionState.SessionState		63

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

clustering.py	
Clustering base class	65
comparison_plots.py	65
dbscan.py	
Implementation of the DBSCAN algorithm	66
dbscan_heuristic.py	
Implementation of DBSCAN parameter estimation heuristic	67
heuristic_web.py	
Webfrontend for the DBSCAN heuristic implemented using streamlit	67
indices.py	
Evaluation Modul to compare clustering results	68
kmeans.py	
Implementation of the k-means algorithm	69
kmedians.py	
Implementation of the k-medians algorithm	69
kmedoids.py	
Implementation of the k-medoids algorithm	70
result_calculation.py	70
results.py	
Handler for saving and loading results	71
SessionState.py	
Taken from https://gist.github.com/tvst/036da038ab3e999a64497f42de966a92	
71	
web_frontend.py	
Webfrontend for project	72

Chapter 6

Namespace Documentation

6.1 clustering Namespace Reference

Classes

- class [Clustering](#)

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

6.2 comparison_plots Namespace Reference

Variables

- list [kalgos](#) = ['kmeans', 'kmedians', 'kmedoids']
- dictionary [kalgoclass](#) = {'kmeans': [kmeansClustering](#), 'kmedians': [kmediansClustering](#), 'kmedoids': [kmedoidsClustering](#)}
- list [distances](#) = ["euclidean", "manhattan", "chebyshev", "cosine"]
- list [datasets](#) = ["iris", "wine", "diabetes", "housevotes"]
- list [index_ext_eval](#) = ["ARI", "NMI", "Completeness Score", "Homogeneity Score"]
- list [index_int_eval](#) = ["Silhouette Score"]
- list [num_of_classes](#) = [3,3,2,2]
- int [seed](#) = 42
- [results](#) = [Results](#)("./results")
- [all_kalgos](#) = np.zeros((3,4, 9,len([index_ext_eval](#))))
- [all_kalgos_int](#) = pd.DataFrame(columns=['k', 'Distance (Silhouette Score)', 'Distance (Clustering)', 'sil_score', 'kalgo'])
- [all_dist](#) = np.zeros((4, 9,len([index_ext_eval](#))))
- [all_k](#) = np.zeros((9,len([index_ext_eval](#))))
- [clusters](#)
- [stuff](#)
- [s](#)
- [c](#)
- [d](#)
- [k](#)
- dictionary [cluster](#) = [kalgoclass](#)[c]([d](#), [s](#), [seed](#))

- `clustered_data` = `np.zeros(len(cluster.data))`
- dictionary `labels` = `cluster.labels.tolist()`
- `predicted` = `clustered_data.tolist()`
- `l1` = `Indices(predicted, labels)`
- `index_scores` = `np.zeros_like(index_ext_eval, dtype=float)`
- `index_score` = `l1.index_internal(index=index_int_eval[0], points=cluster.data.tolist(), metric=di)`
- `index`
- `fig` = `plt.figure(figsize=(15, 10))`
- `bbox_to_anchor`
- `loc`
- `borderaxespad`
- `ax`
- `figsize`
- `data`
- `x`
- `y`
- `hue`
- `style`
- `legend`
- `all_kalgos_df` = `pd.DataFrame(all_kalgos[:, :, num_of_classes[isx]-2, i], columns=distances, index=kalgos)`
- `kind`
- `title`

6.2.1 Variable Documentation

6.2.1.1 `all_dist`

```
comparison_plots.all_dist = np.zeros((4, 9, len(index_ext_eval)))
```

6.2.1.2 `all_k`

```
comparison_plots.all_k = np.zeros((9, len(index_ext_eval)))
```

6.2.1.3 `all_kalgos`

```
comparison_plots.all_kalgos = np.zeros((3, 4, 9, len(index_ext_eval)))
```

6.2.1.4 all_kalgos_df

```
comparison_plots.all_kalgos_df = pd.DataFrame(all_kalgos[:, :, num_of_classes[isx]-2, i], columns=distances,
index=kalgos)
```

6.2.1.5 all_kalgos_int

```
comparison_plots.all_kalgos_int = pd.DataFrame(columns=['k', 'Distance (Silhouette Score)',
'Distance (Clustering)', 'sil_score', 'kalgo'])
```

6.2.1.6 ax

```
comparison_plots.ax
```

6.2.1.7 bbox_to_anchor

```
comparison_plots.bbox_to_anchor
```

6.2.1.8 borderaxespad

```
comparison_plots.borderaxespad
```

6.2.1.9 c

```
comparison_plots.c
```

6.2.1.10 cluster

```
dictionary comparison_plots.cluster = kalgoclass[c](d, s, seed)
```

6.2.1.11 clustered_data

```
comparison_plots.clustered_data = np.zeros(len(cluster.data))
```

6.2.1.12 clusters

```
comparison_plots.clusters
```

6.2.1.13 d

```
comparison_plots.d
```

6.2.1.14 data

```
comparison_plots.data
```

6.2.1.15 datasets

```
list comparison_plots.datasets = ["iris", "wine", "diabetes", "housevotes"]
```

6.2.1.16 distances

```
comparison_plots.distances = ["euclidean", "manhattan", "chebyshev", "cosine"]
```

6.2.1.17 fig

```
comparison_plots.fig = plt.figure(figsize=(15, 10))
```

6.2.1.18 figsize

```
comparison_plots.figsize
```


6.2.1.19 hue

```
comparison_plots.hue
```

6.2.1.20 I1

```
comparison_plots.I1 = Indices(predicted, labels)
```

6.2.1.21 index

```
comparison_plots.index
```

6.2.1.22 index_ext_eval

```
list comparison_plots.index_ext_eval = ["ARI", "NMI", "Completeness Score", "Homogeneity Score"]
```

6.2.1.23 index_int_eval

```
list comparison_plots.index_int_eval = ["Silhouette Score"]
```

6.2.1.24 index_score

```
comparison_plots.index_score = I1.index_internal(index=index_int_eval[0], points=cluster.↵  
data.tolist(), metric=di)
```

6.2.1.25 index_scores

```
comparison_plots.index_scores = np.zeros_like(index_ext_eval, dtype=float)
```

6.2.1.26 k

`comparison_plots.k`

6.2.1.27 kalgoclass

```
dictionary comparison_plots.kalgoclass = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering,  
'kmedoids': kmedoidsClustering}
```

6.2.1.28 kalgos

```
list comparison_plots.kalgos = ['kmeans', 'kmedians', 'kmedoids']
```

6.2.1.29 kind

`comparison_plots.kind`

6.2.1.30 labels

```
dictionary comparison_plots.labels = cluster.labels.tolist()
```

6.2.1.31 legend

`comparison_plots.legend`

6.2.1.32 loc

`comparison_plots.loc`

6.2.1.33 num_of_classes

```
list comparison_plots.num_of_classes = [3,3,2,2]
```

6.2.1.34 predicted

```
comparison_plots.predicted = clustered_data.tolist()
```

6.2.1.35 results

```
comparison_plots.results = Results("./results")
```

6.2.1.36 s

```
comparison_plots.s
```

6.2.1.37 seed

```
int comparison_plots.seed = 42
```

6.2.1.38 stuff

```
comparison_plots.stuff
```

6.2.1.39 style

```
comparison_plots.style
```

6.2.1.40 title

```
comparison_plots.title
```

6.2.1.41 x

`comparison_plots.x`

6.2.1.42 y

`comparison_plots.y`

6.3 dbscan Namespace Reference

Classes

- class [DBSCANClustering](#)
implements DBSCAN Clustering
uses the scikit-learn DBSCAN implementation

6.4 dbscan_heuristic Namespace Reference

Classes

- class [DBSCANHeuristic](#)
implements the DBSCAN heuristic proposed in the original DBSCAN paper:

6.5 heuristic_web Namespace Reference

Variables

- [page_title](#)
- [page_icon](#)
- [key](#)
- [col1](#)
- [col2](#)
- [dataset](#) = `col1.selectbox('Choose a beautiful dataset', ['iris', 'wine', 'diabetes', 'housevotes'])`
- dictionary [cluster_dist_desc](#)
- [cluster_dist](#) = `col1.selectbox('Choose an awesome distance measure', list(cluster_dist_desc.keys()))`
- [k](#) = `col2.slider('Choose a nice value for k', min_value=1, max_value=20, step=1, value=4)`
- [submit_button](#) = `st.form_submit_button(label='Calculate kdist Graph')`
- [heu](#) = [DBSCANHeuristic](#)()
- [kdist](#) = `heu.kdist(k)`
- [reverse](#)
- [df](#)
- [nearest](#) = `alt.selection(type='single', nearest=True, on='mouseover', fields=['points'], empty='none')`
- [yaxis](#) = `alt.Y("dist", axis=alt.Axis(title=f"{k}-dist"))`
- [line](#)
- [selectors](#) = `alt.Chart(df).mark_point().encode(x='points', opacity=alt.value(0)).add_selection(nearest)`
- [points](#) = `line.mark_point(color="red").encode(opacity=alt.condition(nearest, alt.value(1), alt.value(0)))`
- [text](#) = `line.mark_text(aligned='left', dx=5, dy=-5, color="red").encode(text=alt.condition(nearest, "label:N", alt.condition(nearest, "label:N", alt.value('')))).transform_calculate(label=f"distance: " + format(datum.dist, ".2f"))`
- [textp](#) = `line.mark_text(aligned='left', dx=5, dy=-15, color="red").encode(text=alt.condition(nearest, "label:N", alt.value(''))).transform_calculate(label=f"format((1 - (datum.points-1) / {len(kdist)}) * 100, ".2f") + "% core points")`
- [rules](#) = `alt.Chart(df).mark_rule(color='gray').encode(x="points").transform_filter(nearest)`
- [use_container_width](#)

6.5.1 Variable Documentation

6.5.1.1 cluster_dist

```
heuristic_web.cluster_dist = coll.selectbox('Choose an awesome distance measure',list(cluster←
_dist_desc.keys()))
```

6.5.1.2 cluster_dist_desc

```
dictionary heuristic_web.cluster_dist_desc
```

Initial value:

```
1 = {'euclidean': 'd(x,y) = \sqrt{\sum_{i=1}^n (|x_i-y_i|)^2}',
2     'manhattan': 'd(x,y) = \sum\limits_{i=1}^n |x_i - y_i|',
3     'chebyshev': 'd(x,y) = \max(|x_i - y_i|)',
4     'cosine': 'd(x,y) = \frac{\arccos(\frac{\sum_{i=1}^n x_i
y_i}{\sqrt{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2}})}{\pi}'}
```

6.5.1.3 col1

```
heuristic_web.col1
```

6.5.1.4 col2

```
heuristic_web.col2
```

6.5.1.5 dataset

```
heuristic_web.dataset = coll.selectbox('Choose a beautiful dataset',['iris', 'wine', 'diabetes',
'housevotes'])
```

6.5.1.6 df

```
heuristic_web.df
```

Initial value:

```
1 = pd.DataFrame(
2     [[i+1, kdist[i]] for i in range(len(kdist))],
3     columns=["points", "dist"])
```

6.5.1.7 heu

```
heuristic_web.heu = DBSCANHeuristic()
```

6.5.1.8 k

```
heuristic_web.k = col2.slider("Choose a nice value for k", min_value=1, max_value=20, step=1, value=4)
```

6.5.1.9 kdist

```
heuristic_web.kdist = heu.kdist(k)
```

6.5.1.10 key

```
heuristic_web.key
```

6.5.1.11 line

```
heuristic_web.line
```

Initial value:

```
1 = alt.Chart(df).mark_line(point=True).encode(x="points", y=yaxis).properties(  
2     title=f"DBSCAN Heuristic k={k}, {cluster_dist} distance")
```

6.5.1.12 nearest

```
heuristic_web.nearest = alt.selection(type='single', nearest=True, on='mouseover', fields=['points'], empty='none')
```

6.5.1.13 page_icon

```
heuristic_web.page_icon
```

6.5.1.14 page_title

```
heuristic_web.page_title
```

6.5.1.15 points

```
heuristic_web.points = line.mark_point(color="red").encode(opacity=alt.condition(nearest,  
alt.value(1), alt.value(0)))
```

6.5.1.16 reverse

```
heuristic_web.reverse
```

6.5.1.17 rules

```
heuristic_web.rules = alt.Chart(df).mark_rule(color='gray').encode(x="points").transform_↵  
filter(nearest)
```

6.5.1.18 selectors

```
heuristic_web.selectors = alt.Chart(df).mark_point().encode(x='points', opacity=alt.value(0)).add↵  
_selection(nearest)
```

6.5.1.19 submit_button

```
heuristic_web.submit_button = st.form_submit_button(label='Calculate kdist Graph')
```

6.5.1.20 text

```
heuristic_web.text = line.mark_text(align='left', dx=5, dy=-5, color="red").encode(text=alt.↵  
condition(nearest, "label:N", alt.value(' '))).transform_calculate(label=f'"distance: " +  
format(datum.dist, ".2f")')
```

6.5.1.21 textp

```
heuristic_web.textp = line.mark_text(align='left', dx=5, dy=-15, color="red").encode(text=alt.↵
condition(nearest, "label:N", alt.value(' ')).transform_calculate(label=f'format( (1 - (datum.↵
points-1) / {len(kdist)}) * 100, ".2f") + "% core points"')
```

6.5.1.22 use_container_width

```
heuristic_web.use_container_width
```

6.5.1.23 yaxis

```
heuristic_web.yaxis = alt.Y("dist", axis=alt.Axis(title=f"{k}-dist"))
```

6.6 indices Namespace Reference

Classes

- class [Indices](#)
calculates [Indices](#) for computed cluster labels uses the scikit library

6.7 kmeans Namespace Reference

Classes

- class [kmeansClustering](#)
*Class implementing k-Means Clustering
uses the pyclustering k-means implementation
centers can be initialised using the k++ or the random initialiser.*

6.8 kmedians Namespace Reference

Classes

- class [kmediansClustering](#)
implements k-Medians Clustering uses the pyclustering k-medians implementation centers are initialised using the random initialiser

6.9 kmedoids Namespace Reference

Classes

- class [kmedoidsClustering](#)
implements k-Medians Clustering
uses the scikit-learn-extra k-medoids implementation
centers are set using the k++ initialiser if not set differently

6.10 result_calculation Namespace Reference

Variables

- list [kalgos](#) = ['kmeans', 'kmedians', 'kmedoids']
- dictionary [kalgoclass](#) = {'kmeans': [kmeansClustering](#), 'kmedians': [kmediansClustering](#), 'kmedoids': [kmedoidsClustering](#)}
- list [distances](#) = ["euclidean", "manhattan", "chebyshev", "cosine"]
- list [datasets](#) = ["iris", "wine", "diabetes", "housevotes"]
- int [seed](#) = 42
- [results](#) = [Results](#)("./results")
- [s](#)
- [c](#)
- [d](#)
- [k](#)
- dictionary [alg](#) = [kalgoclass](#)[[c](#)]([d](#), [s](#), [seed](#))
- [clusters](#)
- [centers](#)
- [minpts](#)
- [m](#)
- [eps](#)

6.10.1 Variable Documentation

6.10.1.1 alg

```
result_calculation.alg = kalgoclass[c](d, s, seed)
```

6.10.1.2 c

```
result_calculation.c
```

6.10.1.3 centers

```
result_calculation.centers
```

6.10.1.4 clusters

```
result_calculation.clusters
```

6.10.1.5 d

```
result_calculation.d
```

6.10.1.6 datasets

```
list result_calculation.datasets = ["iris", "wine", "diabetes", "housevotes"]
```

6.10.1.7 distances

```
list result_calculation.distances = ["euclidean", "manhattan", "chebyshev", "cosine"]
```

6.10.1.8 eps

```
result_calculation.eps
```

6.10.1.9 k

```
result_calculation.k
```

6.10.1.10 kalgoclass

```
dictionary result_calculation.kalgoclass = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering,  
'kmedoids': kmedoidsClustering}
```

6.10.1.11 kalgos

```
list result_calculation.kalgos = ['kmeans', 'kmedians', 'kmedoids']
```

6.10.1.12 m

```
result_calculation.m
```

6.10.1.13 minpts

```
result_calculation.minpts
```

6.10.1.14 results

```
result_calculation.results = Results("./results")
```

6.10.1.15 s

```
result_calculation.s
```

6.10.1.16 seed

```
int result_calculation.seed = 42
```

6.11 results Namespace Reference

Classes

- class [Results](#)

class for easily saving and loading already calculated clustering results

every dataset has a folder containing subfolders for every clustering algorithm containing more subfolders for every distance measure.

6.12 SessionState Namespace Reference

Classes

- class [SessionState](#)

Functions

- def [get](#) (**kwargs)
Gets a [SessionState](#) object for the current session.

6.12.1 Function Documentation

6.12.1.1 [get\(\)](#)

```
def SessionState.get (
    ** kwargs )
```

Gets a [SessionState](#) object for the current session.

Creates a new object if necessary.

Parameters

kwargs : any
Default values you want to add to the session state, if we're creating a new one.

Example

```
>>> session_state = get(user_name='', favorite_color='black')
>>> session_state.user_name
''
>>> session_state.user_name = 'Mary'
>>> session_state.favorite_color
'black'
```

Since you set user_name above, next time your script runs this will be the result:

```
>>> session_state = get(user_name='', favorite_color='black')
>>> session_state.user_name
'Mary'
```

6.13 web_frontend Namespace Reference

Variables

- `page_title`
- `page_icon`
- `session_state = SessionState.get(indices_data=pd.DataFrame())`
- `seeded = st.checkbox('Use precalculated results (with random seed for reproduction).', value=True)`
- `seed = None`
- `seaplots = st.checkbox('Use interactive charts', value=True)`
- `resulthandler = Results("./code/results")`
- `col1`
- `col2`
- `dataset = col1.selectbox('Choose a beautiful dataset', ['iris', 'wine', 'diabetes', 'housevotes'])`
- dictionary `cluster_dist_desc`
- `cluster_dist = col1.selectbox('Choose an awesome distance measure', list(cluster_dist_desc.keys()))`
- dictionary `cluster_algo_class = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering, 'kmedoids': kmedoidsClustering, 'DBSCAN': DBSCANClustering}`
- `cluster_algo = col2.selectbox('Choose a lovely clustering algorithm', list(cluster_algo_class.keys()))`
- dictionary `cluster = cluster_algo_class[cluster_algo](cluster_dist, dataset, seed)`
- `epsilon = col2.slider("Choose a nice value for epsilon", min_value=0.1, max_value=20.0, step=0.1)`
- `minpts = col2.slider("Choose a minimal number of nearest points", min_value=1, max_value=20, step=1, value=5)`
- `eps`
- `clusters`
- `stuff`
- `k_value = col2.slider("Choose a nice value for k (number of clusters)", min_value=2, max_value=10, step=1, value=3)`
- `k`
- `clustered_data = np.zeros(len(cluster.data))`
- list `color_palette = ['black'] + sns.color_palette("husl", len(set(clustered_data))-1)`
- `weights`
- `perp = col1.slider("Perplexity for t-SNE", 5, 50, 25)`
- `marking_centroids = np.ones(cluster.data.shape[0])`
- `cluster_label = alt.Tooltip("c", title="Cluster ID")`
- `point_label = alt.Tooltip("i", title="Point ID")`
- `dfclusterdata = pd.DataFrame()`
- `projected_data_tsne = TSNE(random_state=42, perplexity=perp).fit_transform(cluster.data)`
- `fig`
- `ax`
- `x`
- `y`
- `hue`
- `palette`
- `legend`
- `False`
- `style`
- `size`
- `markers`
- `xaxis = alt.X("xt", axis=alt.Axis(title=None))`
- `yaxis = alt.Y("yt", axis=alt.Axis(title=None))`
- `altcolor = alt.Color("c", legend=None, scale=alt.Scale(domain=[0, 1 if max(clustered_data) == 0 else max(clustered_data)], scheme="turbo"))`

- `tsnealt = alt.Chart(dfclusterdata).mark_circle().encode(x=xaxis, y=yaxis, tooltip=[cluster_label, point_label], color=altcolor).interactive()`
- `use_container_width`
- `projected_data_pca = PCA(random_state=42, n_components=2).fit_transform(cluster.data)`
- `pcaalt = alt.Chart(dfclusterdata).mark_circle().encode(x=xaxis, y=yaxis, tooltip=[cluster_label, point_label], color=altcolor).interactive()`
- `dataexpander = st.beta_expander("data")`
- `add_result = col1.button('Add')`
- `reset_tmp = col2.button('Reset')`
- `indices_data`
- `string val = "epsilon="+str(epsilon)+" , np="+str(minpts)`
- `df = session_state.indices_data`
- `dictionary labels = cluster.labels.tolist()`
- `predicted = clustered_data.tolist()`
- `list precalc = []`
- `list index_eval = ["ARI", "NMI", "Completeness Score", "Homogeneity Score", "Silhouette Score"]`
- `l1 = Indices(predicted, labels)`
- `score = l1.index_external(index_eval[i])`
- `list datasets = []`
- `list results = [[1, "maximum reference value"]]`
- `list desc_list = []`
- `desc = np.array(desc_list)`
- `stats = np.zeros(len(results))`
- `ys`
- `xs = np.arange(len(labels))`
- `float width = 0.5`
- `align`
- `color`
- `angles = np.linspace(0, 2*np.pi, len(desc), endpoint=False)`
- `subplot_kw`
- `linewidth`
- `alpha`

6.13.1 Variable Documentation

6.13.1.1 add_result

```
web_frontend.add_result = col1.button('Add')
```

6.13.1.2 align

```
web_frontend.align
```

6.13.1.3 alpha

`web_frontend.alpha`

6.13.1.4 altcolor

```
web_frontend.altcolor = alt.Color("c", legend=None, scale=alt.Scale(domain=[0, 1 if max(clustered_data)
== 0 else max(clustered_data)], scheme="turbo"))
```

6.13.1.5 angles

```
web_frontend.angles = np.linspace(0, 2*np.pi, len(desc), endpoint=False)
```

6.13.1.6 ax

`web_frontend.ax`

6.13.1.7 cluster

```
dictionary web_frontend.cluster = cluster_algo_class[cluster_algo](cluster_dist, dataset,
seed)
```

6.13.1.8 cluster_algo

```
web_frontend.cluster_algo = col2.selectbox('Choose a lovely clustering algorithm', list(cluster_↵
_algo_class.keys()))
```

6.13.1.9 cluster_algo_class

```
dictionary web_frontend.cluster_algo_class = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering,
'kmedoids': kmedoidsClustering, 'DBSCAN': DBSCANClustering}
```

6.13.1.10 cluster_dist

```
web_frontend.cluster_dist = coll.selectbox('Choose an awesome distance measure', list(cluster←
_dist_desc.keys()))
```

6.13.1.11 cluster_dist_desc

```
dictionary web_frontend.cluster_dist_desc
```

Initial value:

```
1 = {'euclidean': 'd(x,y) = \sqrt{\sum_{i=1}^n (|x_i-y_i|)^2}',
2     'manhattan': 'd(x,y) = \sum\limits_{i=1}^n |x_i - y_i|',
3     'chebyshev': 'd(x,y) = \max(|x_i - y_i|)',
4     'cosine': 'd(x,y) = \frac{\arccos(\frac{\sum_{i=1}^n x_i
y_i}{\sqrt{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2}})}{\pi}'}
```

6.13.1.12 cluster_label

```
web_frontend.cluster_label = alt.Tooltip("c", title="Cluster ID")
```

6.13.1.13 clustered_data

```
web_frontend.clustered_data = np.zeros(len(cluster.data))
```

6.13.1.14 clusters

```
web_frontend.clusters
```

6.13.1.15 col1

```
web_frontend.col1
```

6.13.1.16 col2

```
web_frontend.col2
```


6.13.1.17 color

```
web_frontend.color
```

6.13.1.18 color_palette

```
list web_frontend.color_palette = ['black'] + sns.color_palette("husl", len(set(clustered_data))-1)
```

6.13.1.19 dataexpander

```
web_frontend.dataexpander = st.beta_expander("data")
```

6.13.1.20 dataset

```
web_frontend.dataset = coll.selectbox('Choose a beautiful dataset',['iris', 'wine', 'diabetes',  
'housevotes'])
```

6.13.1.21 datasets

```
list web_frontend.datasets = []
```

6.13.1.22 desc

```
web_frontend.desc = np.array(desc_list)
```

6.13.1.23 desc_list

```
list web_frontend.desc_list = []
```

6.13.1.24 df

```
web_frontend.df = session_state.indices_data
```

6.13.1.25 dfclusterdata

```
web_frontend.dfclusterdata = pd.DataFrame()
```

6.13.1.26 eps

```
web_frontend.eps
```

6.13.1.27 epsilon

```
web_frontend.epsilon = col2.slider("Choose a nice value for epsilon", min_value=0.1, max_value=20.0, step=0.1)
```

6.13.1.28 False

```
web_frontend.False
```

6.13.1.29 fig

```
web_frontend.fig
```

6.13.1.30 hue

```
web_frontend.hue
```

6.13.1.31 I1

```
web_frontend.I1 = Indices(predicted, labels)
```

6.13.1.32 index_eval

```
web_frontend.index_eval = ["ARI", "NMI", "Completeness Score", "Homogeneity Score", "Silhouette Score"]
```

6.13.1.33 indices_data

```
web_frontend.indices_data
```

6.13.1.34 k

```
web_frontend.k
```

6.13.1.35 k_value

```
web_frontend.k_value = col2.slider("Choose a nice value for k (number of clusters)", min_value=2, max_value=10, step=1, value=3)
```

6.13.1.36 labels

```
web_frontend.labels = cluster.labels.tolist()
```

6.13.1.37 legend

```
web_frontend.legend
```

6.13.1.38 linewidth

```
web_frontend.linewidth
```

6.13.1.39 markers

```
web_frontend.markers
```

6.13.1.40 marking_centroids

```
web_frontend.marking_centroids = np.ones(cluster.data.shape[0])
```

6.13.1.41 minpts

```
web_frontend.minpts = col2.slider("Choose a minimal number of nearest points", min_value=1,  
max_value=20, step=1, value=5)
```

6.13.1.42 page_icon

```
web_frontend.page_icon
```

6.13.1.43 page_title

```
web_frontend.page_title
```

6.13.1.44 palette

```
web_frontend.palette
```

6.13.1.45 pcaalt

```
web_frontend.pcaalt = alt.Chart(dfclusterdata).mark_circle().encode(x=xaxis, y=yaxis, tooltip=[cluster_label, point_label], color=altcolor).interactive()
```

6.13.1.46 perp

```
web_frontend.perp = coll.slider("Perplexity for t-SNE", 5, 50, 25)
```

6.13.1.47 point_label

```
web_frontend.point_label = alt.Tooltip("i", title="Point ID")
```

6.13.1.48 precalc

```
list web_frontend.precalc = []
```

6.13.1.49 predicted

```
web_frontend.predicted = clustered_data.tolist()
```

6.13.1.50 projected_data_pca

```
web_frontend.projected_data_pca = PCA(random_state=42, n_components=2).fit_transform(cluster.<->data)
```

6.13.1.51 projected_data_tsne

```
web_frontend.projected_data_tsne = TSNE(random_state=42, perplexity=perp).fit_transform(cluster.<->data)
```

6.13.1.52 reset_tmp

```
web_frontend.reset_tmp = col2.button('Reset')
```

6.13.1.53 resulthandler

```
web_frontend.resulthandler = Results("./code/results")
```

6.13.1.54 results

```
list web_frontend.results = [[1, "maximum reference value"]]
```

6.13.1.55 score

```
web_frontend.score = I1.index_external(index_eval[i])
```

6.13.1.56 seaplots

```
web_frontend.seaplots = st.checkbox('Use interactive charts', value=True)
```

6.13.1.57 seed

```
int web_frontend.seed = None
```

6.13.1.58 seeded

```
web_frontend.seeded = st.checkbox('Use precalculated results (with random seed for reproduction).',  
value=True)
```

6.13.1.59 session_state

```
web_frontend.session_state = SessionState.get(indices_data=pd.DataFrame())
```

6.13.1.60 size

```
web_frontend.size
```

6.13.1.61 stats

```
web_frontend.stats = np.zeros(len(results))
```

6.13.1.62 stuff

```
web_frontend.stuff
```

6.13.1.63 style

```
web_frontend.style
```

6.13.1.64 subplot_kw

```
web_frontend.subplot_kw
```

6.13.1.65 tsnealt

```
web_frontend.tsnealt = alt.Chart(dfclusterdata).mark_circle().encode(x=xaxis, y=yaxis, tooltip=[cluster_label, point_label], color=altcolor).interactive()
```

6.13.1.66 use_container_width

```
web_frontend.use_container_width
```

6.13.1.67 val

```
string web_frontend.val = "epsilon="+str(epsilon)+"", np="+str(minpts)
```

6.13.1.68 weights

```
web_frontend.weights
```

6.13.1.69 width

```
web_frontend.width = 0.5
```

6.13.1.70 x

```
web_frontend.x
```

6.13.1.71 xaxis

```
web_frontend.xaxis = alt.X("xt", axis=alt.Axis(title=None))
```

6.13.1.72 xs

```
web_frontend.xs = np.arange(len(labels))
```

6.13.1.73 y

```
web_frontend.y
```

6.13.1.74 yaxis

```
web_frontend.yaxis = alt.Y("yt", axis=alt.Axis(title=None))
```

6.13.1.75 ys

```
web_frontend.ys
```

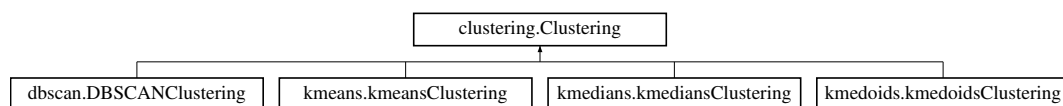

Chapter 7

Class Documentation

7.1 clustering.Clustering Class Reference

Base 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, seed=None)`
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
- [seed](#)
seed for initializer, None if no seed is used
- [datadf](#)
dataset as pandas frame.

7.1.1 Detailed Description

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

7.1.2 Constructor & Destructor Documentation

7.1.2.1 `__init__()`

```
def clustering.Clustering.__init__ (
    self,
    metric,
    dataset,
    seed = None )
```

constructor

Parameters

<i>metric</i>	metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine"
<i>dataset</i>	dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes"

Reimplemented in [kmedoids.kmedoidsClustering](#), [kmedians.kmediansClustering](#), [kmeans.kmeansClustering](#), and [dbscan.DBSCANClustering](#).

7.1.3 Member Function Documentation

7.1.3.1 cluster()

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

does nothing in the meta class.

needs to be implemented in the inheriting cluster algorithm classes

7.1.3.2 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

<i>path</i>	filepath to the dataset
<i>skip</i>	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

7.1.3.3 load_data()

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

loads in a dataset, standardises it and sets it as self.data attribute

7.1.3.4 pyc_metric()

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

returns a distance metric which is usable by the pyclustering algorithms

Parameters

<i>distance</i>	metric string. allowed: "euclidean", "manhattan", "chebyshev", "cosine"
-----------------	---

Returns

pyclustering distance_metric object, None when distance is not supported

7.1.4 Member Data Documentation

7.1.4.1 data

`clustering.Clustering.data`

data that gets clustered

7.1.4.2 datadf

`clustering.Clustering.datadf`

dataset as pandas frame.

needed for web frontend later

7.1.4.3 dataset

`clustering.Clustering.dataset`

dataset name as string

7.1.4.4 labels

`clustering.Clustering.labels`

expected cluster values

7.1.4.5 metric

`clustering.Clustering.metric`

metric name as string or pyclustering distance_metric object

7.1.4.6 seed

`clustering.Clustering.seed`

seed for initializer, None if no seed is used

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

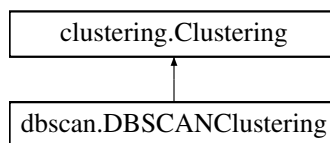
- [clustering.py](#)

7.2 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, seed=None)`
constructor, seed can be given but is not used.
- `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

7.2.1 Detailed Description

implements DBSCAN Clustering
uses the scikit-learn DBSCAN implementation

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `__init__()`

```
def dbscan.DBSCANClustering.__init__ (
    self,
    metric,
    dataset,
    seed = None )
```

constructor, seed can be given but is not used.

its passing is allowed to simplify the code in the web frontend

Parameters

<i>metric</i>	metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine"
<i>dataset</i>	dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes"

Reimplemented from [clustering.Clustering](#).

7.2.3 Member Function Documentation

7.2.3.1 `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

<i>eps</i>	Distance for the Eps-Neighbourhood
<i>minPts</i>	Minmal number of points in a cluster

Returns

formatted clustered data

7.2.3.2 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

<i>labels</i>	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

7.2.4 Member Data Documentation**7.2.4.1 data**

```
dbscan.DBSCANClustering.data
```

data that gets clustered

7.2.4.2 dataset

```
dbscan.DBSCANClustering.dataset
```

dataset name as string

7.2.4.3 labels

```
dbscan.DBSCANClustering.labels
```

expected cluster values

7.2.4.4 metric

`dbscan.DBSCANClustering.metric`

metric name as string

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

- [dbscan.py](#)

7.3 dbscan_heuristic.DBSCANHeuristic Class Reference

implements the DBSCAN heuristic proposed in the original DBSCAN paper:

Public Member Functions

- `def __init__ (self)`
constructor.
- `def set_metric (self, metric)`
setter for the metric
- `def set_dataset (self, dataset)`
sets and loads the dataset using the DBSCANClustering objects `load_data()` function
- `def kdist (self, k)`
calculates all k -distances for the dataset
- `def plot_kdist (self, kdist)`
plots the sorted $kdist$ graph using `matplotlib`

Public Attributes

- [clustering](#)
DBSCANClustering object for loading datasets.
- [metric](#)
metric as string ("euclidean", "cosine", "manhattan", "chebyshev")
- [k](#)
variable k used in the k -dist calculation

7.3.1 Detailed Description

implements the DBSCAN heuristic proposed in the original DBSCAN paper:

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, page 226–231. AAAI Press, 1996.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `__init__()`

```
def dbscan_heuristic.DBSCANHeuristic.__init__ (
    self )
```

constructor.

uses a DBSCANClustering object for loading the datasets

7.3.3 Member Function Documentation

7.3.3.1 `kdist()`

```
def dbscan_heuristic.DBSCANHeuristic.kdist (
    self,
    k )
```

calculates all k-distances for the dataset

Parameters

<i>k</i>	variable for the k-dist. Natural Number.
----------	--

7.3.3.2 `plot_kdist()`

```
def dbscan_heuristic.DBSCANHeuristic.plot_kdist (
    self,
    kdist )
```

plots the sorted kdist graph using matplotlib

Parameters

<i>k-dist</i>	list containing the k-distances for every point of the dataset
---------------	--

7.3.3.3 `set_dataset()`

```
def dbscan_heuristic.DBSCANHeuristic.set_dataset (
    self,
    dataset )
```

sets and loads the dataset using the DBSCANClustering objects `load_data()` function

Parameters

<i>dataset</i>	string with name of the dataset used ("iris", "wine", "diabetes", "housevotes")
----------------	---

7.3.3.4 set_metric()

```
def dbscan_heuristic.DBSCANHeuristic.set_metric (
    self,
    metric )
```

setter for the metric

Parameters

<i>metric</i>	string containing name of the metric ("euclidean", "cosine", "manhattan", "chebyshev")
---------------	--

7.3.4 Member Data Documentation**7.3.4.1 clustering**

```
dbscan_heuristic.DBSCANHeuristic.clustering
```

DBSCANClustering object for loading datasets.

7.3.4.2 k

```
dbscan_heuristic.DBSCANHeuristic.k
```

variable k used in the k-dist calculation

7.3.4.3 metric

```
dbscan_heuristic.DBSCANHeuristic.metric
```

metric as string ("euclidean", "cosine", "manhattan", "chebyshev")

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

- [dbscan_heuristic.py](#)

7.4 indices.Indices Class Reference

calculates [Indices](#) for computed cluster labels uses the scikit library

Public Member Functions

- `def __init__ (self, cluster_calc, cluster_label)`
constructor
- `def index_external (self, index)`
Function to calculate external index scores
ARI, AMI, Homogeneity Score and Completeness Score @params index string with name of index used ("ARI", "AMI", "Homogeneity Score", "Completeness Score")
- `def index_internal (self, index, points, metric)`
Function to calculate internal index scores

Public Attributes

- [cluster_calc](#)
calculated clustering results
- [cluster_label](#)
expected cluster results

7.4.1 Detailed Description

calculates [Indices](#) for computed cluster labels uses the scikit library

7.4.2 Constructor & Destructor Documentation

7.4.2.1 __init__()

```
def indices.Indices.__init__ (
    self,
    cluster_calc,
    cluster_label )
```

constructor

Parameters

<i>cluster_calc</i>	calculated clustering results
<i>cluster_label</i>	expected cluster results

7.4.3 Member Function Documentation

7.4.3.1 `index_external()`

```
def indices.Indices.index_external (
    self,
    index )
```

Function to calculate external index scores

ARI, AMI, Homogeneity Score and Completeness Score @params index string with name of index used ("ARI", "AMI", "Homogeneity Score", "Completeness Score")

7.4.3.2 `index_internal()`

```
def indices.Indices.index_internal (
    self,
    index,
    points,
    metric )
```

Function to calculate internal index scores

Parameters

<i>index</i>	
<i>points</i>	
<i>metric</i>	

7.4.4 Member Data Documentation

7.4.4.1 `cluster_calc`

```
indices.Indices.cluster_calc
```

calculated clustering results

7.4.4.2 cluster_label

`indices.Indices.cluster_label`

expected cluster results

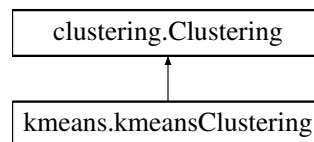
The documentation for this class was generated from the following file:

- [indices.py](#)

7.5 kmeans.kmeansClustering Class Reference

Class implementing k-Means Clustering
uses the pylustering 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, seed=None)`
constructor
- `def cluster (self, k, plusplus=True)`
clustering method.

Public Attributes

- [metric](#)
metric name as pylustering distance_metric object
- [dataset](#)
dataset name as string
- [data](#)
data that gets clustered
- [labels](#)
expected cluster values
- [seed](#)
seed for initializer, None if no seed is used

7.5.1 Detailed Description

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

7.5.2 Constructor & Destructor Documentation

7.5.2.1 `__init__()`

```
def kmeans.kmeansClustering.__init__ (
    self,
    metric,
    dataset,
    seed = None )
```

constructor

Parameters

<i>metric</i>	metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine"
<i>dataset</i>	dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes"

Reimplemented from [clustering.Clustering](#).

7.5.3 Member Function Documentation

7.5.3.1 `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

<i>k</i>	number of clusters that are generated
<i>plusplus</i>	will use k++ initialiser if true

Returns

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

7.5.4 Member Data Documentation

7.5.4.1 data

`kmeans.kmeansClustering.data`

data that gets clustered

7.5.4.2 dataset

`kmeans.kmeansClustering.dataset`

dataset name as string

7.5.4.3 labels

`kmeans.kmeansClustering.labels`

expected cluster values

7.5.4.4 metric

`kmeans.kmeansClustering.metric`

metric name as pyclustering distance_metric object

7.5.4.5 seed

`kmeans.kmeansClustering.seed`

seed for initializer, None if no seed is used

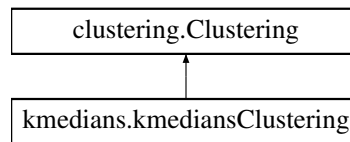
The documentation for this class was generated from the following file:

- [kmeans.py](#)

7.6 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, seed=None)`
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
- [seed](#)
seed for initializer, None if no seed is used

7.6.1 Detailed Description

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

7.6.2 Constructor & Destructor Documentation

7.6.2.1 __init__()

```
def kmedians.kmediansClustering.__init__ (
    self,
    metric,
    dataset,
    seed = None )
```

constructor

Parameters

<i>metric</i>	metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine"
<i>dataset</i>	dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes"

Reimplemented from [clustering.Clustering](#).

7.6.3 Member Function Documentation

7.6.3.1 cluster()

```
def kmedians.kmediansClustering.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

<i>k</i>	number of clusters that are generated
----------	---------------------------------------

Returns

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

7.6.4 Member Data Documentation

7.6.4.1 data

```
kmedians.kmediansClustering.data
```

data that gets clustered

7.6.4.2 dataset

```
kmedians.kmediansClustering.dataset
```

dataset name as string

7.6.4.3 labels

`kmedians.kmediansClustering.labels`

expected cluster values

7.6.4.4 metric

`kmedians.kmediansClustering.metric`

metric name as pyclustering distance_metric object

7.6.4.5 seed

`kmedians.kmediansClustering.seed`

seed for initializer, None if no seed is used

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

- [kmedians.py](#)

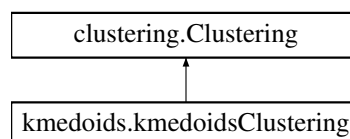
7.7 kmedoids.kmedoidsClustering Class Reference

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, seed=None)`
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
- [seed](#)
seed for initializer, None if no seed is used

7.7.1 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

7.7.2 Constructor & Destructor Documentation

7.7.2.1 __init__()

```
def kmedoids.kmedoidsClustering.__init__ (
    self,
    metric,
    dataset,
    seed = None )
```

constructor

Parameters

<i>metric</i>	metric description as string. allowed: "euclidean", "manhattan", "chebyshev", "cosine"
<i>dataset</i>	dataset given as string. allowed: "diabetes", "iris", "wine", "housevotes"

Reimplemented from [clustering.Clustering](#).

7.7.3 Member Function Documentation

7.7.3.1 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

<i>k</i>	number of clusters that are generated
<i>init</i>	initialisation parameter. Default: "k-medoids++"

Returns

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

7.7.3.2 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

<i>labels</i>	labels returned from the KMedoids algorithm
---------------	---

Returns

clusters formatted similarly to the pyclustering algorithms

7.7.4 Member Data Documentation

7.7.4.1 data

```
kmedoids.kmedoidsClustering.data
```

data that gets clustered

7.7.4.2 dataset

`kmedoids.kmedoidsClustering.dataset`

dataset name as string

7.7.4.3 labels

`kmedoids.kmedoidsClustering.labels`

expected cluster values

7.7.4.4 metric

`kmedoids.kmedoidsClustering.metric`

metric name as string

7.7.4.5 seed

`kmedoids.kmedoidsClustering.seed`

seed for initializer, None if no seed is used

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

- [kmedoids.py](#)

7.8 results.Results Class Reference

class for easily saving and loading already calculated clustering results

every dataset has a folder containing subfolders for every clustering algorithm containing more subfolders for every distance measure.

Public Member Functions

- `def __init__ (self, parentpath)`
constructor.
- `def get_path (self, dataset, algorithm, metric, **kwargs)`
builds and returns the filepath to the json file fitting the given parameters
- `def set_exists (self, dataset, algorithm, metric, **kwargs)`
checks if a file for a result defined by the parameters exists
- `def load_set (self, dataset, algorithm, metric, **kwargs)`
loads results fitting the given parameters from a json file
- `def save_set (self, dataset, algorithm, metric, clusters, centers, **kwargs)`
saves cluster results in a json file

Public Attributes

- `parent`

7.8.1 Detailed Description

class for easily saving and loading already calculated clustering results

every dataset has a folder containing subfolders for every clustering algorithm containing more subfolders for every distance measure.

Clustering results are saved as json files in their respective folders.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 __init__()

```
def results.Results.__init__ (
    self,
    parentpath )
```

constructor.

needs the filepath to the parent directory where the json files are supposed to be saved

Parameters

<code>parentpath</code>	filepath to the parent directory
-------------------------	----------------------------------

7.8.3 Member Function Documentation

7.8.3.1 get_path()

```
def results.Results.get_path (
    self,
    dataset,
    algorithm,
    metric,
    ** kwargs )
```

builds and returns the filepath to the json file fitting the given parameters

Parameters

<i>dataset</i>	string with the name of the dataset ("iris", "wine", "diabetes", "DBSCAN")
<i>algorithm</i>	string with the name of the algorithm ("kmeans", "kmedians", "kmedoids", "DBSCAN")
<i>metric</i>	string with the name of the distance measure ("euclidean", "cosine", "chebyshev", "manhattan")
<i>**kwargs</i>	algorithm specific parameters. Needs to be either "k" or "minpts" and "eps"

Returns

filepath to the correct json file

7.8.3.2 load_set()

```
def results.Results.load_set (
    self,
    dataset,
    algorithm,
    metric,
    ** kwargs )
```

loads results fitting the given parameters from a json file

Parameters

<i>dataset</i>	string with the name of the dataset ("iris", "wine", "diabetes", "DBSCAN")
<i>algorithm</i>	string with the name of the algorithm ("kmeans", "kmedians", "kmedoids", "DBSCAN")
<i>metric</i>	string with the name of the distance measure ("euclidean", "cosine", "chebyshev", "manhattan")
<i>**kwargs</i>	algorithm specific parameters. Needs to be either "k" or "minpts" and "eps"

Returns

loaded clustering results (clusters and centers)

7.8.3.3 save_set()

```
def results.Results.save_set (
    self,
    dataset,
    algorithm,
    metric,
    clusters,
    centers,
    ** kwargs )
```

saves cluster results in a json file

Parameters

<i>dataset</i>	string with the name of the dataset ("iris", "wine", "diabetes", "DBSCAN")
<i>algorithm</i>	string with the name of the algorithm ("kmeans", "kmedians", "kmedoids", "DBSCAN")
<i>metric</i>	string with the name of the distance measure ("euclidean", "cosine", "chebyshev", "manhattan")
<i>**kwargs</i>	algorithm specific parameters. Needs to be either "k" or "minpts" and "eps"

7.8.3.4 set_exists()

```
def results.Results.set_exists (
    self,
    dataset,
    algorithm,
    metric,
    ** kwargs )
```

checks if a file for a result defined by the parameters exists

Parameters

<i>dataset</i>	string with the name of the dataset ("iris", "wine", "diabetes", "DBSCAN")
<i>algorithm</i>	string with the name of the algorithm ("kmeans", "kmedians", "kmedoids", "DBSCAN")
<i>metric</i>	string with the name of the distance measure ("euclidean", "cosine", "chebyshev", "manhattan")
<i>**kwargs</i>	algorithm specific parameters. Needs to be either "k" or "minpts" and "eps"

Returns

True if file exists, False if not

7.8.4 Member Data Documentation

7.8.4.1 parent

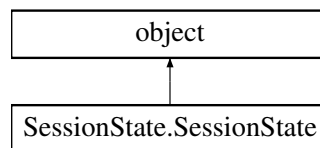
```
results.Results.parent
```

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

- [results.py](#)

7.9 SessionState.SessionState Class Reference

Inheritance diagram for SessionState.SessionState:



Public Member Functions

- `def __init__(self, **kwargs)`
A new [SessionState](#) object.

7.9.1 Constructor & Destructor Documentation

7.9.1.1 __init__()

```
def SessionState.SessionState.__init__ (
    self,
    ** kwargs )
```

A new [SessionState](#) object.

```
Parameters
-----
**kwargs : any
    Default values for the session state.

Example
-----
>>> session_state = SessionState(user_name='', favorite_color='black')
>>> session_state.user_name = 'Mary'
''
>>> session_state.favorite_color
'black'
```

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

- [SessionState.py](#)

Chapter 8

File Documentation

8.1 clustering.py File Reference

contains the clustering base class

Classes

- class [clustering.Clustering](#)
*Base Class for all subsequent clustering algorithms
implements all functions needed for running the different
cluster algorithms.*

Namespaces

- [clustering](#)

8.1.1 Detailed Description

contains the clustering base class

8.2 comparison_plots.py File Reference

Namespaces

- [comparison_plots](#)

Variables

- list `comparison_plots.kalgos` = ['kmeans', 'kmedians', 'kmedoids']
- dictionary `comparison_plots.kalgoclass` = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering, 'kmedoids': kmedoidsClustering}
- list `comparison_plots.distances` = ["euclidean", "manhattan", "chebyshev", "cosine"]
- list `comparison_plots.datasets` = ["iris", "wine", "diabetes", "housevotes"]
- list `comparison_plots.index_ext_eval` = ["ARI", "NMI", "Completeness Score", "Homogeneity Score"]
- list `comparison_plots.index_int_eval` = ["Silhouette Score"]
- list `comparison_plots.num_of_classes` = [3,3,2,2]
- int `comparison_plots.seed` = 42
- `comparison_plots.results` = Results("./results")
- `comparison_plots.all_kalgos` = np.zeros((3,4, 9,len(index_ext_eval)))
- `comparison_plots.all_kalgos_int` = pd.DataFrame(columns=['k', 'Distance (Silhouette Score)', 'Distance (Clustering)', 'sil_score', 'kalgo'])
- `comparison_plots.all_dist` = np.zeros((4, 9,len(index_ext_eval)))
- `comparison_plots.all_k` = np.zeros((9,len(index_ext_eval)))
- `comparison_plots.clusters`
- `comparison_plots.stuff`
- `comparison_plots.s`
- `comparison_plots.c`
- `comparison_plots.d`
- `comparison_plots.k`
- dictionary `comparison_plots.cluster` = kalgoclass[c](d, s, seed)
- `comparison_plots.clustered_data` = np.zeros(len(cluster.data))
- dictionary `comparison_plots.labels` = cluster.labels.tolist()
- `comparison_plots.predicted` = clustered_data.tolist()
- `comparison_plots.l1` = Indices(predicted, labels)
- `comparison_plots.index_scores` = np.zeros_like(index_ext_eval, dtype=float)
- `comparison_plots.index_score` = l1.index_internal(index=index_int_eval[0], points=cluster.data.tolist(), metric=di)
- `comparison_plots.index`
- `comparison_plots.fig` = plt.figure(figsize=(15, 10))
- `comparison_plots.bbox_to_anchor`
- `comparison_plots.loc`
- `comparison_plots.borderaxespadd`
- `comparison_plots.ax`
- `comparison_plots.figsize`
- `comparison_plots.data`
- `comparison_plots.x`
- `comparison_plots.y`
- `comparison_plots.hue`
- `comparison_plots.style`
- `comparison_plots.legend`
- `comparison_plots.all_kalgos_df` = pd.DataFrame(all_kalgos[:, :, num_of_classes[isx]-2, i], columns=distances, index=kalgos)
- `comparison_plots.kind`
- `comparison_plots.title`

8.3 dbscan.py File Reference

implementation of the DBSCAN algorithm.

Classes

- class [dbscan.DBSCANClustering](#)
implements DBSCAN Clustering
uses the scikit-learn DBSCAN implementation

Namespaces

- [dbscan](#)

8.3.1 Detailed Description

implementation of the DBSCAN algorithm.

8.4 dbscan_heuristic.py File Reference

implementation of DBSCAN parameter estimation heuristic

Classes

- class [dbscan_heuristic.DBSCANHeuristic](#)
implements the DBSCAN heuristic proposed in the original DBSCAN paper:

Namespaces

- [dbscan_heuristic](#)

8.4.1 Detailed Description

implementation of DBSCAN parameter estimation heuristic

8.5 heuristic_web.py File Reference

webfrontend for the DBSCAN heuristic implemented using streamlit.

Namespaces

- [heuristic_web](#)

Variables

- [heuristic_web.page_title](#)
- [heuristic_web.page_icon](#)
- [heuristic_web.key](#)
- [heuristic_web.col1](#)
- [heuristic_web.col2](#)
- [heuristic_web.dataset](#) = col1.selectbox('Choose a beautiful dataset',['iris', 'wine', 'diabetes', 'housevotes'])
- dictionary [heuristic_web.cluster_dist_desc](#)
- [heuristic_web.cluster_dist](#) = col1.selectbox('Choose an awesome distance measure',list(cluster_dist_desc.keys()))
- [heuristic_web.k](#) = col2.slider("Choose a nice value for k", min_value=1, max_value=20, step=1, value=4)
- [heuristic_web.submit_button](#) = st.form_submit_button(label='Calculate kdist Graph')
- [heuristic_web.heu](#) = DBSCANHeuristic()
- [heuristic_web.kdist](#) = heu.kdist(k)
- [heuristic_web.reverse](#)
- [heuristic_web.df](#)
- [heuristic_web.nearest](#) = alt.selection(type='single', nearest=True, on='mouseover', fields=['points'], empty='none')
- [heuristic_web.yaxis](#) = alt.Y("dist", axis=alt.Axis(title=f"{k}-dist"))
- [heuristic_web.line](#)
- [heuristic_web.selectors](#) = alt.Chart(df).mark_point().encode(x='points', opacity=alt.value(0)).add_selection(nearest)
- [heuristic_web.points](#) = line.mark_point(color="red").encode(opacity=alt.condition(nearest, alt.value(1), alt.value(0)))
- [heuristic_web.text](#) = line.mark_text(aligned='left', dx=5, dy=-5, color="red").encode(text=alt.condition(nearest, "label:N", alt.value(' '))).transform_calculate(label=f"distance: " + format(datum.dist, ".2f"))
- [heuristic_web.textp](#) = line.mark_text(aligned='left', dx=5, dy=-15, color="red").encode(text=alt.condition(nearest, "label:N", alt.value(' '))).transform_calculate(label=f"format((1 - (datum.points-1) / {len(kdist)}) * 100, ".2f") + "% core points")
- [heuristic_web.rules](#) = alt.Chart(df).mark_rule(color='gray').encode(x="points").transform_filter(nearest)
- [heuristic_web.use_container_width](#)

8.5.1 Detailed Description

webfrontend for the DBSCAN heuristic implemented using streamlit.

uses altair charts for displaying the chart

8.6 indices.py File Reference

Evaluation Modul to compare clustering results.

Classes

- class [indices.Indices](#)
calculates [Indices](#) for computed cluster labels uses the scikit library

Namespaces

- [indices](#)

8.6.1 Detailed Description

Evaluation Modul to compare clustering results.

8.7 kmeans.py File Reference

implementation of the k-means algorithm.

Classes

- class [kmeans.kmeansClustering](#)
*Class implementing k-Means Clustering
uses the pyclustering k-means implementation
centers can be initialised using the k++ or the random initialiser.*

Namespaces

- [kmeans](#)

8.7.1 Detailed Description

implementation of the k-means algorithm.

8.8 kmedians.py File Reference

implementation of the k-medians algorithm.

Classes

- class [kmedians.kmediansClustering](#)
implements k-Medians Clustering uses the pyclustering k-medians implementation centers are initialised using the random initialiser

Namespaces

- [kmedians](#)

8.8.1 Detailed Description

implementation of the k-medians algorithm.

8.9 kmedoids.py File Reference

implementation of the k-medoids algorithm.

Classes

- class [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

Namespaces

- [kmedoids](#)

8.9.1 Detailed Description

implementation of the k-medoids algorithm.

8.10 result_calculation.py File Reference

Namespaces

- [result_calculation](#)

Variables

- list [result_calculation.kalgos](#) = ['kmeans', 'kmedians', 'kmedoids']
- dictionary [result_calculation.kalgoclass](#) = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering, 'kmedoids': kmedoidsClustering}
- list [result_calculation.distances](#) = ["euclidean", "manhattan", "chebyshev", "cosine"]
- list [result_calculation.datasets](#) = ["iris", "wine", "diabetes", "housevotes"]
- int [result_calculation.seed](#) = 42
- [result_calculation.results](#) = Results("./results")
- [result_calculation.s](#)
- [result_calculation.c](#)
- [result_calculation.d](#)
- [result_calculation.k](#)
- dictionary [result_calculation.alg](#) = kalgoclass[c](d, s, seed)
- [result_calculation.clusters](#)
- [result_calculation.centers](#)
- [result_calculation.minpts](#)
- [result_calculation.m](#)
- [result_calculation.eps](#)

8.11 results.py File Reference

handler for saving and loading results.

Classes

- class [results.Results](#)
class for easily saving and loading already calculated clustering results

every dataset has a folder containing subfolders for every clustering algorithm containing more subfolders for every distance measure.

Namespaces

- [results](#)

8.11.1 Detailed Description

handler for saving and loading results.

8.12 SessionState.py File Reference

taken from <https://gist.github.com/tvst/036da038ab3e999a64497f42de966a92>.

Classes

- class [SessionState.SessionState](#)

Namespaces

- [SessionState](#)

Functions

- def [SessionState.get](#) (**kwargs)
Gets a [SessionState](#) object for the current session.

8.12.1 Detailed Description

taken from <https://gist.github.com/tvst/036da038ab3e999a64497f42de966a92>.

Please refer to this gist and its original author

Hack to add per-session state to Streamlit.

8.12.1.1 Usage

```
import SessionState
session_state = SessionState.get(user_name="", favorite_color='black') session_↵
_state.user_name

"

session_state.user_name = 'Mary' session_state.favorite_color

'black'
```

Since you set user_name above, next time your script runs this will be the result:

```
session_state = get(user_name="", favorite_color='black') session_state.user_↵
name

'Mary'
```

8.13 web_frontend.py File Reference

webfrontend for project.

Namespaces

- [web_frontend](#)

Variables

- [web_frontend.page_title](#)
- [web_frontend.page_icon](#)
- [web_frontend.session_state](#) = [SessionState.get](#)(indices_data=pd.DataFrame())
- [web_frontend.seeded](#) = st.checkbox("Use precalculated results (with random seed for reproduction).", value=True)
- [web_frontend.seed](#) = None
- [web_frontend.seaplots](#) = st.checkbox("Use interactive charts", value=True)
- [web_frontend.resulthandler](#) = Results("./code/results")
- [web_frontend.col1](#)
- [web_frontend.col2](#)
- [web_frontend.dataset](#) = col1.selectbox('Choose a beautiful dataset',['iris', 'wine', 'diabetes', 'housevotes'])
- dictionary [web_frontend.cluster_dist_desc](#)
- [web_frontend.cluster_dist](#) = col1.selectbox('Choose an awesome distance measure', list(cluster_dist_desc.↵ keys()))
- dictionary [web_frontend.cluster_algo_class](#) = {'kmeans': kmeansClustering, 'kmedians': kmediansClustering, 'kmedoids': kmedoidsClustering, 'DBSCAN': DBSCANClustering}
- [web_frontend.cluster_algo](#) = col2.selectbox('Choose a lovely clustering algorithm', list(cluster_algo_class.↵ keys()))
- dictionary [web_frontend.cluster](#) = cluster_algo_class[cluster_algo](cluster_dist, dataset, seed)

- `web_frontend.epsilon` = `col2.slider("Choose a nice value for epsilon", min_value=0.1, max_value=20, step=0.1)`
- `web_frontend.minpts` = `col2.slider("Choose a minimal number of nearest points", min_value=1, max_value=20, step=1, value=5)`
- `web_frontend.eps`
- `web_frontend.clusters`
- `web_frontend.stuff`
- `web_frontend.k_value` = `col2.slider("Choose a nice value for k (number of clusters)", min_value=2, max_value=10, step=1, value=3)`
- `web_frontend.k`
- `web_frontend.clustered_data` = `np.zeros(len(cluster.data))`
- list `web_frontend.color_palette` = `['black'] + sns.color_palette("husl", len(set(clustered_data))-1)`
- `web_frontend.weights`
- `web_frontend.perp` = `col1.slider("Perplexity for t-SNE", 5, 50, 25)`
- `web_frontend.marking_centroids` = `np.ones(cluster.data.shape[0])`
- `web_frontend.cluster_label` = `alt.Tooltip("c", title="Cluster ID")`
- `web_frontend.point_label` = `alt.Tooltip("i", title="Point ID")`
- `web_frontend.dfclusterdata` = `pd.DataFrame()`
- `web_frontend.projected_data_tsne` = `TSNE(random_state=42, perplexity=perp).fit_transform(cluster.data)`
- `web_frontend.fig`
- `web_frontend.ax`
- `web_frontend.x`
- `web_frontend.y`
- `web_frontend.hue`
- `web_frontend.palette`
- `web_frontend.legend`
- `web_frontend.False`
- `web_frontend.style`
- `web_frontend.size`
- `web_frontend.markers`
- `web_frontend.xaxis` = `alt.X("xt", axis=alt.Axis(title=None))`
- `web_frontend.yaxis` = `alt.Y("yt", axis=alt.Axis(title=None))`
- `web_frontend.altcolor` = `alt.Color("c", legend=None, scale=alt.Scale(domain=[0, 1 if max(clustered_data) == 0 else max(clustered_data)], scheme="turbo"))`
- `web_frontend.tsnealt` = `alt.Chart(dfclusterdata).mark_circle().encode(x=xaxis, y=yaxis, tooltip=[cluster_label, point_label], color=altcolor).interactive()`
- `web_frontend.use_container_width`
- `web_frontend.projected_data_pca` = `PCA(random_state=42, n_components=2).fit_transform(cluster.data)`
- `web_frontend.pcaalt` = `alt.Chart(dfclusterdata).mark_circle().encode(x=xaxis, y=yaxis, tooltip=[cluster_label, point_label], color=altcolor).interactive()`
- `web_frontend.dataexpander` = `st.beta_expander("data")`
- `web_frontend.add_result` = `col1.button('Add')`
- `web_frontend.reset_tmp` = `col2.button('Reset')`
- `web_frontend.indices_data`
- string `web_frontend.val` = `"epsilon="+str(epsilon)+"", np="+str(minpts)`
- `web_frontend.df` = `session_state.indices_data`
- dictionary `web_frontend.labels` = `cluster.labels.tolist()`
- `web_frontend.predicted` = `clustered_data.tolist()`
- list `web_frontend.precalc` = `[]`
- list `web_frontend.index_eval` = `["ARI", "NMI", "Completeness Score", "Homogeneity Score", "Silhouette Score"]`
- `web_frontend.l1` = `Indices(predicted, labels)`
- `web_frontend.score` = `l1.index_external(index_eval[i])`
- list `web_frontend.datasets` = `[]`
- list `web_frontend.results` = `[[1, "maximum reference value"]]`

- list `web_frontend.desc_list` = []
- `web_frontend.desc` = `np.array(desc_list)`
- `web_frontend.stats` = `np.zeros(len(results))`
- `web_frontend.ys`
- `web_frontend.xs` = `np.arange(len(labels))`
- float `web_frontend.width` = 0.5
- `web_frontend.align`
- `web_frontend.color`
- `web_frontend.angles` = `np.linspace(0, 2*np.pi, len(desc), endpoint=False)`
- `web_frontend.subplot_kw`
- `web_frontend.linewidth`
- `web_frontend.alpha`

8.13.1 Detailed Description

webfrontend for project.

implemented using streamlit. displays parameter selection, clustering results, data and the evaluation module.
Charts can be displayed using seaborn or altair

8.14 `/home/nordegraf/Uni/8.Semester/DataScience1/group42/README.md` File Reference ↩

Index

/home/nordegraf/Uni/8.Semester/DataScienceI/datascience1_group42/README.md, 40
74
__init__
 clustering.Clustering, 40
 dbscan.DBSCANClustering, 44
 dbscan_heuristic.DBSCANHeuristic, 46
 indices.Indices, 49
 kmeans.kmeansClustering, 52
 kmedians.kmediansClustering, 54
 kmedoids.kmedoidsClustering, 57
 results.Results, 60
 SessionState.SessionState, 63
add_result
 web_frontend, 28
alg
 result_calculation, 23
align
 web_frontend, 28
all_dist
 comparison_plots, 12
all_k
 comparison_plots, 12
all_kalgos
 comparison_plots, 12
all_kalgos_df
 comparison_plots, 12
all_kalgos_int
 comparison_plots, 13
alpha
 web_frontend, 28
altcolor
 web_frontend, 29
angles
 web_frontend, 29
ax
 comparison_plots, 13
 web_frontend, 29
bbox_to_anchor
 comparison_plots, 13
borderaxespad
 comparison_plots, 13
c
 comparison_plots, 13
 result_calculation, 23
centers
 result_calculation, 23
cluster
 comparison_plots, 13
 dbscan.DBSCANClustering, 44
 kmeans.kmeansClustering, 52
 kmedians.kmediansClustering, 55
 kmedoids.kmedoidsClustering, 57
 web_frontend, 29
cluster_algo
 web_frontend, 29
cluster_algo_class
 web_frontend, 29
cluster_calc
 indices.Indices, 50
cluster_dist
 heuristic_web, 19
 web_frontend, 29
cluster_dist_desc
 heuristic_web, 19
 web_frontend, 30
cluster_label
 indices.Indices, 50
 web_frontend, 30
clustered_data
 comparison_plots, 13
 web_frontend, 30
clustering, 11
 dbscan_heuristic.DBSCANHeuristic, 48
clustering.Clustering, 39
 __init__, 40
 cluster, 40
 data, 42
 datadf, 42
 dataset, 42
 house_load, 41
 labels, 42
 load_data, 41
 metric, 42
 pyc_metric, 41
 seed, 43
clustering.py, 65
clusters
 comparison_plots, 14
 result_calculation, 24
 web_frontend, 30
col1
 heuristic_web, 19
 web_frontend, 30
col2
 heuristic_web, 19

- web_frontend, 30
- color
 - web_frontend, 30
- color_palette
 - web_frontend, 31
- comparison_plots, 11
 - all_dist, 12
 - all_k, 12
 - all_kalgos, 12
 - all_kalgos_df, 12
 - all_kalgos_int, 13
 - ax, 13
 - bbox_to_anchor, 13
 - borderaxespad, 13
 - c, 13
 - cluster, 13
 - clustered_data, 13
 - clusters, 14
 - d, 14
 - data, 14
 - datasets, 14
 - distances, 14
 - fig, 14
 - figsize, 14
 - hue, 14
 - l1, 15
 - index, 15
 - index_ext_eval, 15
 - index_int_eval, 15
 - index_score, 15
 - index_scores, 15
 - k, 15
 - kalgoclass, 16
 - kalgos, 16
 - kind, 16
 - labels, 16
 - legend, 16
 - loc, 16
 - num_of_classes, 16
 - predicted, 17
 - results, 17
 - s, 17
 - seed, 17
 - stuff, 17
 - style, 17
 - title, 17
 - x, 17
 - y, 18
- comparison_plots.py, 65
- d
 - comparison_plots, 14
 - result_calculation, 24
- data
 - clustering.Clustering, 42
 - comparison_plots, 14
 - dbscan.DBSCANClustering, 45
 - kmeans.kmeansClustering, 52
 - kmedians.kmediansClustering, 55
 - kmedoids.kmedoidsClustering, 58
- datadf
 - clustering.Clustering, 42
- dataexpander
 - web_frontend, 31
- dataset
 - clustering.Clustering, 42
 - dbscan.DBSCANClustering, 45
 - heuristic_web, 19
 - kmeans.kmeansClustering, 53
 - kmedians.kmediansClustering, 55
 - kmedoids.kmedoidsClustering, 58
 - web_frontend, 31
- datasets
 - comparison_plots, 14
 - result_calculation, 24
 - web_frontend, 31
- dbscan, 18
- dbscan.DBSCANClustering, 43
 - __init__, 44
 - cluster, 44
 - data, 45
 - dataset, 45
 - labels, 45
 - metric, 45
 - package, 45
- dbscan.py, 66
- dbscan_heuristic, 18
- dbscan_heuristic.DBSCANHeuristic, 46
 - __init__, 46
 - clustering, 48
 - k, 48
 - kdist, 47
 - metric, 48
 - plot_kdist, 47
 - set_dataset, 47
 - set_metric, 48
- dbscan_heuristic.py, 67
- desc
 - web_frontend, 31
- desc_list
 - web_frontend, 31
- df
 - heuristic_web, 19
 - web_frontend, 31
- dfclusterdata
 - web_frontend, 32
- distances
 - comparison_plots, 14
 - result_calculation, 24
- eps
 - result_calculation, 24
 - web_frontend, 32
- epsilon
 - web_frontend, 32
- False
 - web_frontend, 32

- fig
 - comparison_plots, 14
 - web_frontend, 32
- figsize
 - comparison_plots, 14
- get
 - SessionState, 26
- get_path
 - results.Results, 60
- heu
- heuristic_web, 18
 - cluster_dist, 19
 - cluster_dist_desc, 19
 - col1, 19
 - col2, 19
 - dataset, 19
 - df, 19
 - heu, 19
 - k, 20
 - kdist, 20
 - key, 20
 - line, 20
 - nearest, 20
 - page_icon, 20
 - page_title, 20
 - points, 21
 - reverse, 21
 - rules, 21
 - selectors, 21
 - submit_button, 21
 - text, 21
 - textp, 21
 - use_container_width, 22
 - yaxis, 22
- heuristic_web.py, 67
- house_load
 - clustering.Clustering, 41
- hue
 - comparison_plots, 14
 - web_frontend, 32
- l1
 - comparison_plots, 15
 - web_frontend, 32
- index
 - comparison_plots, 15
- index_eval
 - web_frontend, 33
- index_ext_eval
 - comparison_plots, 15
- index_external
 - indices.Indices, 50
- index_int_eval
 - comparison_plots, 15
- index_internal
 - indices.Indices, 50
- index_score
 - comparison_plots, 15
- index_scores
 - comparison_plots, 15
- indices, 22
- indices.Indices, 49
 - __init__, 49
 - cluster_calc, 50
 - cluster_label, 50
 - index_external, 50
 - index_internal, 50
- indices.py, 68
- indices_data
 - web_frontend, 33
- k
 - comparison_plots, 15
 - dbscan_heuristic.DBSCANHeuristic, 48
 - heuristic_web, 20
 - result_calculation, 24
 - web_frontend, 33
- k_value
 - web_frontend, 33
- kalgoclass
 - comparison_plots, 16
 - result_calculation, 24
- kalgos
 - comparison_plots, 16
 - result_calculation, 24
- kdist
 - dbscan_heuristic.DBSCANHeuristic, 47
 - heuristic_web, 20
- key
 - heuristic_web, 20
- kind
 - comparison_plots, 16
- kmeans, 22
- kmeans.kmeansClustering, 51
 - __init__, 52
 - cluster, 52
 - data, 52
 - dataset, 53
 - labels, 53
 - metric, 53
 - seed, 53
- kmeans.py, 69
- kmedians, 22
- kmedians.kmediansClustering, 54
 - __init__, 54
 - cluster, 55
 - data, 55
 - dataset, 55
 - labels, 55
 - metric, 56
 - seed, 56
- kmedians.py, 69
- kmedoids, 23
- kmedoids.kmedoidsClustering, 56
 - __init__, 57

- cluster, 57
- data, 58
- dataset, 58
- labels, 59
- metric, 59
- package, 58
- seed, 59
- kmedoids.py, 70
- labels
 - clustering.Clustering, 42
 - comparison_plots, 16
 - dbscan.DBSCANClustering, 45
 - kmeans.kmeansClustering, 53
 - kmedians.kmediansClustering, 55
 - kmedoids.kmedoidsClustering, 59
 - web_frontend, 33
- legend
 - comparison_plots, 16
 - web_frontend, 33
- line
 - heuristic_web, 20
- linewidth
 - web_frontend, 33
- load_data
 - clustering.Clustering, 41
- load_set
 - results.Results, 61
- loc
 - comparison_plots, 16
- m
 - result_calculation, 25
- markers
 - web_frontend, 34
- marking_centroids
 - web_frontend, 34
- metric
 - clustering.Clustering, 42
 - dbscan.DBSCANClustering, 45
 - dbscan_heuristic.DBSCANHeuristic, 48
 - kmeans.kmeansClustering, 53
 - kmedians.kmediansClustering, 56
 - kmedoids.kmedoidsClustering, 59
- minpts
 - result_calculation, 25
 - web_frontend, 34
- nearest
 - heuristic_web, 20
- num_of_classes
 - comparison_plots, 16
- package
 - dbscan.DBSCANClustering, 45
 - kmedoids.kmedoidsClustering, 58
- page_icon
 - heuristic_web, 20
 - web_frontend, 34
- page_title
 - heuristic_web, 20
 - web_frontend, 34
- palette
 - web_frontend, 34
- parent
 - results.Results, 62
- pcaalt
 - web_frontend, 34
- perp
 - web_frontend, 35
- plot_kdist
 - dbscan_heuristic.DBSCANHeuristic, 47
- point_label
 - web_frontend, 35
- points
 - heuristic_web, 21
- precalc
 - web_frontend, 35
- predicted
 - comparison_plots, 17
 - web_frontend, 35
- projected_data_pca
 - web_frontend, 35
- projected_data_tsne
 - web_frontend, 35
- pyc_metric
 - clustering.Clustering, 41
- reset_tmp
 - web_frontend, 35
- result_calculation, 23
 - alg, 23
 - c, 23
 - centers, 23
 - clusters, 24
 - d, 24
 - datasets, 24
 - distances, 24
 - eps, 24
 - k, 24
 - kalgoclass, 24
 - kalgos, 24
 - m, 25
 - minpts, 25
 - results, 25
 - s, 25
 - seed, 25
- result_calculation.py, 70
- resulthandler
 - web_frontend, 36
- results, 25
 - comparison_plots, 17
 - result_calculation, 25
 - web_frontend, 36
- results.py, 71
- results.Results, 59
 - __init__, 60
 - get_path, 60

- load_set, [61](#)
- parent, [62](#)
- save_set, [61](#)
- set_exists, [62](#)
- reverse
 - heuristic_web, [21](#)
- rules
 - heuristic_web, [21](#)
- s
 - comparison_plots, [17](#)
 - result_calculation, [25](#)
- save_set
 - results.Results, [61](#)
- score
 - web_frontend, [36](#)
- seaplots
 - web_frontend, [36](#)
- seed
 - clustering.Clustering, [43](#)
 - comparison_plots, [17](#)
 - kmeans.kmeansClustering, [53](#)
 - kmedians.kmediansClustering, [56](#)
 - kmedoids.kmedoidsClustering, [59](#)
 - result_calculation, [25](#)
 - web_frontend, [36](#)
- seeded
 - web_frontend, [36](#)
- selectors
 - heuristic_web, [21](#)
- session_state
 - web_frontend, [36](#)
- SessionState, [26](#)
 - get, [26](#)
- SessionState.py, [71](#)
- SessionState.SessionState, [63](#)
 - __init__, [63](#)
- set_dataset
 - dbscan_heuristic.DBSCANHeuristic, [47](#)
- set_exists
 - results.Results, [62](#)
- set_metric
 - dbscan_heuristic.DBSCANHeuristic, [48](#)
- size
 - web_frontend, [37](#)
- stats
 - web_frontend, [37](#)
- stuff
 - comparison_plots, [17](#)
 - web_frontend, [37](#)
- style
 - comparison_plots, [17](#)
 - web_frontend, [37](#)
- submit_button
 - heuristic_web, [21](#)
- subplot_kw
 - web_frontend, [37](#)
- text
 - heuristic_web, [21](#)
 - texttp
 - heuristic_web, [21](#)
 - title
 - comparison_plots, [17](#)
 - tsnealt
 - web_frontend, [37](#)
- use_container_width
 - heuristic_web, [22](#)
 - web_frontend, [37](#)
- val
 - web_frontend, [38](#)
- web_frontend, [27](#)
 - add_result, [28](#)
 - align, [28](#)
 - alpha, [28](#)
 - altcolor, [29](#)
 - angles, [29](#)
 - ax, [29](#)
 - cluster, [29](#)
 - cluster_algo, [29](#)
 - cluster_algo_class, [29](#)
 - cluster_dist, [29](#)
 - cluster_dist_desc, [30](#)
 - cluster_label, [30](#)
 - clustered_data, [30](#)
 - clusters, [30](#)
 - col1, [30](#)
 - col2, [30](#)
 - color, [30](#)
 - color_palette, [31](#)
 - dataexpander, [31](#)
 - dataset, [31](#)
 - datasets, [31](#)
 - desc, [31](#)
 - desc_list, [31](#)
 - df, [31](#)
 - dfclusterdata, [32](#)
 - eps, [32](#)
 - epsilon, [32](#)
 - False, [32](#)
 - fig, [32](#)
 - hue, [32](#)
 - l1, [32](#)
 - index_eval, [33](#)
 - indices_data, [33](#)
 - k, [33](#)
 - k_value, [33](#)
 - labels, [33](#)
 - legend, [33](#)
 - linewidth, [33](#)
 - markers, [34](#)
 - marking_centroids, [34](#)
 - minpts, [34](#)
 - page_icon, [34](#)
 - page_title, [34](#)

- palette, [34](#)
- pcaalt, [34](#)
- perp, [35](#)
- point_label, [35](#)
- precalc, [35](#)
- predicted, [35](#)
- projected_data_pca, [35](#)
- projected_data_tsne, [35](#)
- reset_tmp, [35](#)
- resulthandler, [36](#)
- results, [36](#)
- score, [36](#)
- seaplots, [36](#)
- seed, [36](#)
- seeded, [36](#)
- session_state, [36](#)
- size, [37](#)
- stats, [37](#)
- stuff, [37](#)
- style, [37](#)
- subplot_kw, [37](#)
- tsnealt, [37](#)
- use_container_width, [37](#)
- val, [38](#)
- weights, [38](#)
- width, [38](#)
- x, [38](#)
- xaxis, [38](#)
- xs, [38](#)
- y, [38](#)
- yaxis, [38](#)
- ys, [38](#)
- web_frontend.py, [72](#)
- weights
 - web_frontend, [38](#)
- width
 - web_frontend, [38](#)
- x
 - comparison_plots, [17](#)
 - web_frontend, [38](#)
- xaxis
 - web_frontend, [38](#)
- xs
 - web_frontend, [38](#)
- y
 - comparison_plots, [18](#)
 - web_frontend, [38](#)
- yaxis
 - heuristic_web, [22](#)
 - web_frontend, [38](#)
- ys
 - web_frontend, [38](#)