Gesture Recognition Reading Muscle Activity

Project Phase 3 - Dimensionality Reduction, Clustering and Evaluation

DS5230 / Spring 2024 Semester

Team Members: Liyang Song

Apr 20, 2024

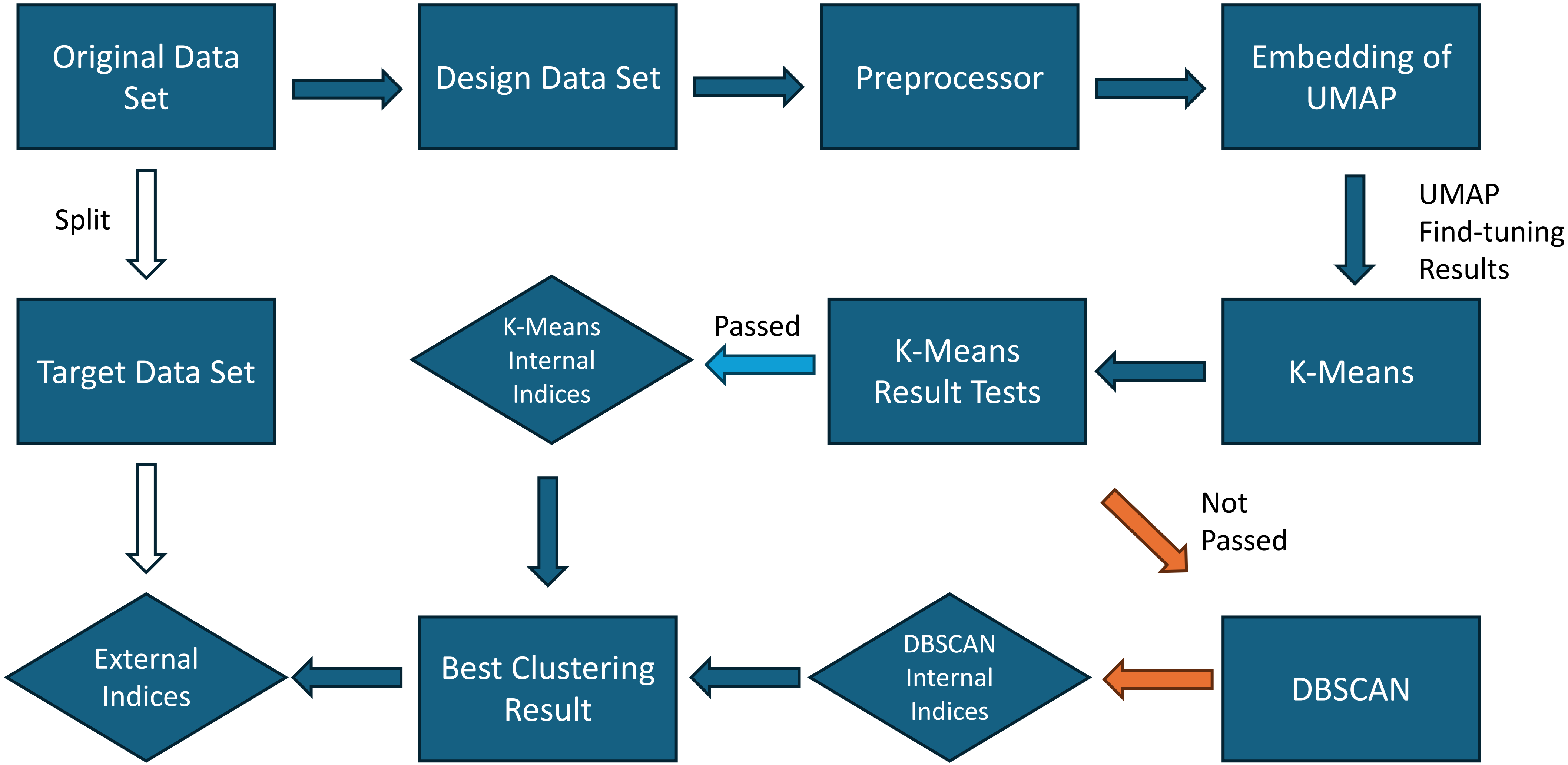
Reproducibility

All works are completed within Jupyter Notebooks and have been fully executed.

To reproduce:

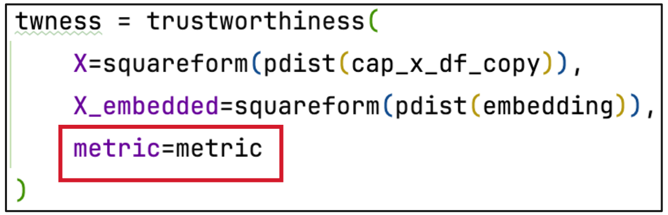
- Download the **environment.yml** and recreate **project_venv** conda environment.
- Download supported python modules.
  - **processing.py**
  - **clustering.py**
  - **validation.py**
- Download Jupyter Notebooks.
  - **project_phase_3_clustering_pipeline.ipynb**
  - **project_phase_3_optimization.ipynb** (takes super long time to execute)
  - **project_phase_3_afd.ipynb** (takes long time to execute)
  - **project_phase_3_validation.ipynb**
- Open notebooks in browser, click **restart the kernel and rerun all cells** button to reproduce all works.
- HTML versions of the fully executed Jupyter Notebooks have also been provided.

Clustering Pipeline

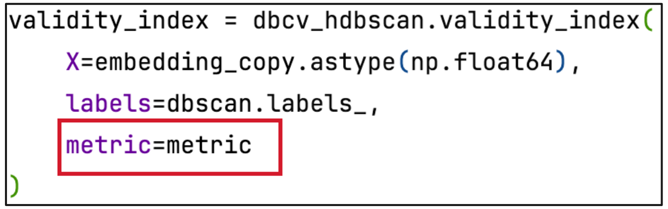


Clustering Pipeline: Updates

- This part extends the clustering pipeline from **assignment_4** and the **midterm exam**, with some errors fixed:
  - The **metric** parameter of **Trustworthiness** is now the same as **UMAP**.
  - The **metric** parameter of **Validity Index** is now the same as **DBSCAN**.

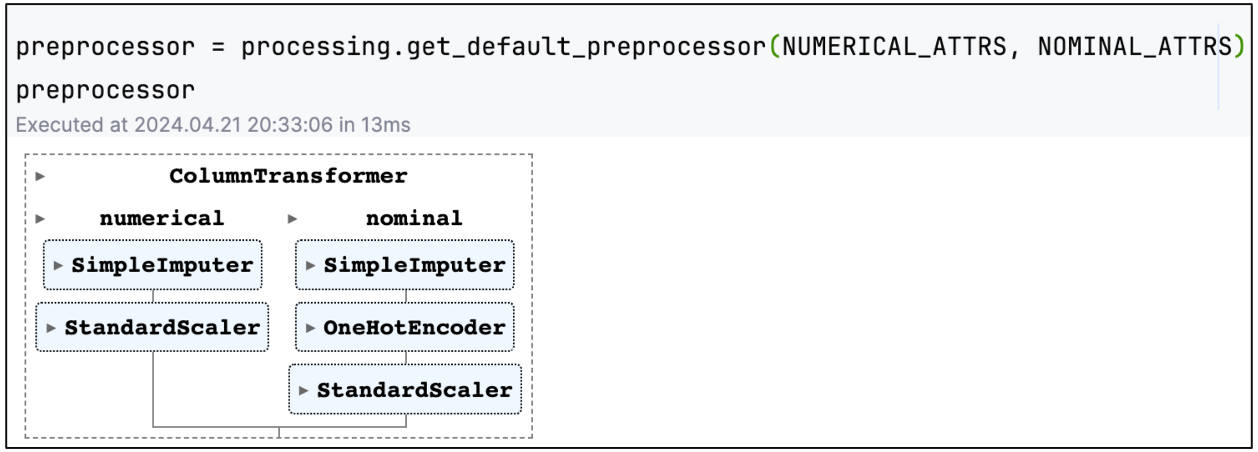


```
twness = trustworthiness(
    X=squareform(pdist(cap_x_df_copy)),
    X_embedded=squareform(pdist(embedding)),
    metric=metric
)
```



```
validity_index = dbcv_hdbscan.validity_index(
    X=embedding_copy.astype(np.float64),
    labels=dbscan.labels_,
    metric=metric
)
```

- Preprocessors such as imputers and scalers are added.

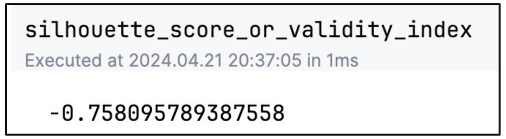


```
preprocessor = processing.get_default_preprocessor(NUMERICAL_ATTRS, NOMINAL_ATTRS)
preprocessor
```

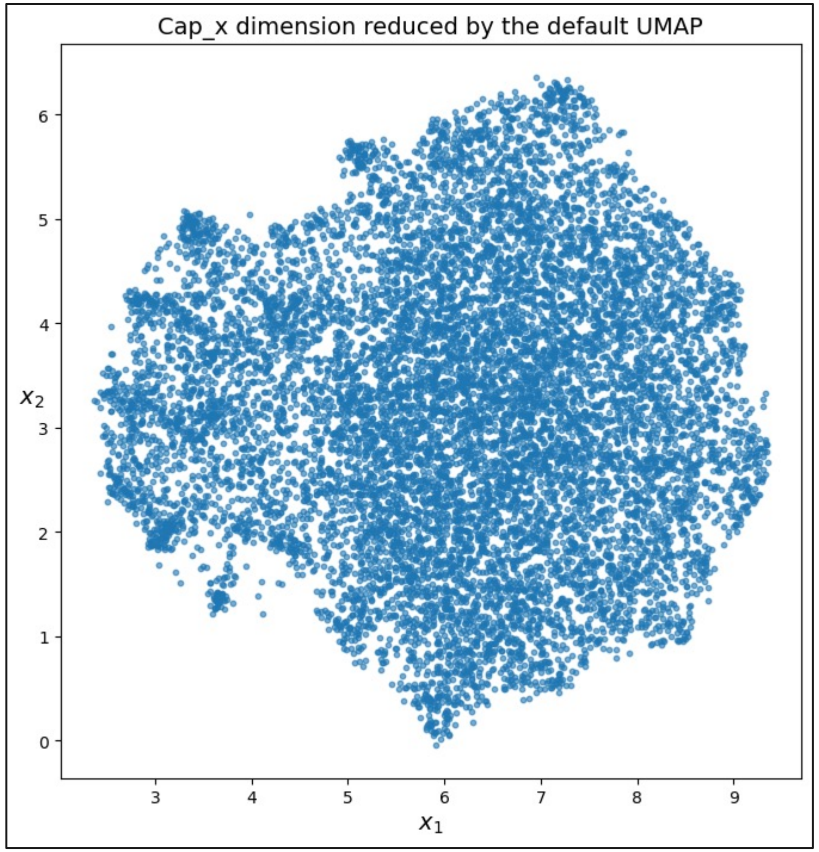
Executed at 2024.04.21 20:33:06 in 13ms

Clustering Pipeline: Default Result

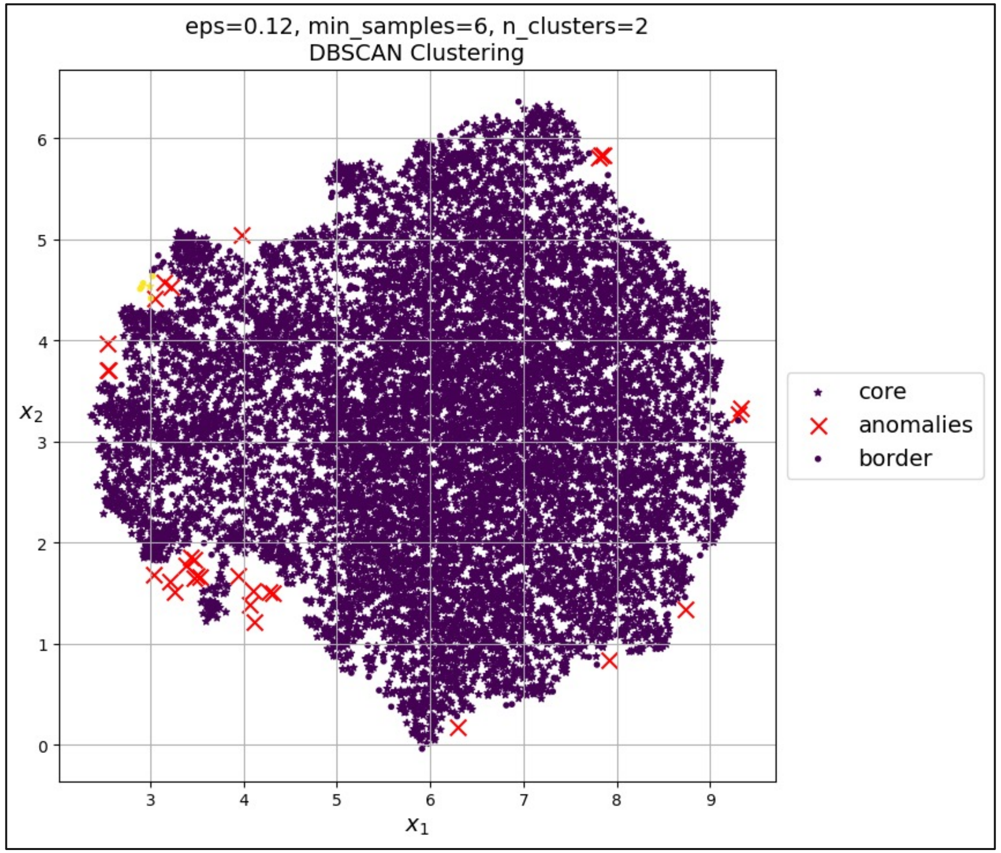
- In the pipeline notebook, we tested the clustering with all default parameters of UMAP and K-means algorithm. This result is clearly not good, but it can at least be used as a baseline for the following pipeline optimization and validation.



```
silhouette_score_or_validity_index
Executed at 2024.04.21 20:37:05 in 1ms

  -0.758095789387558
```



Cap_x dimension reduced by the default UMAP

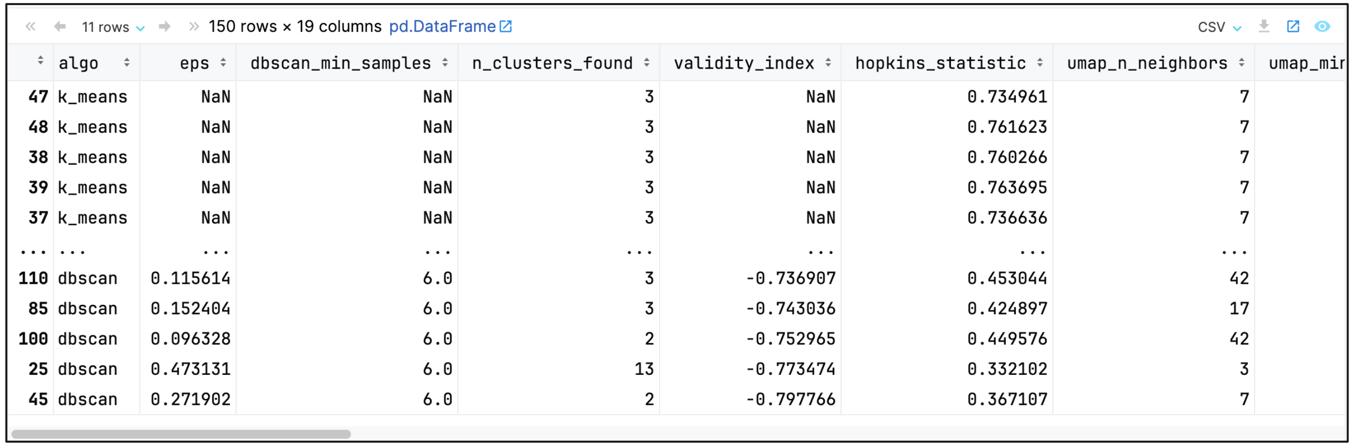


eps=0.12, min_samples=6, n_clusters=2
DBSCAN Clustering

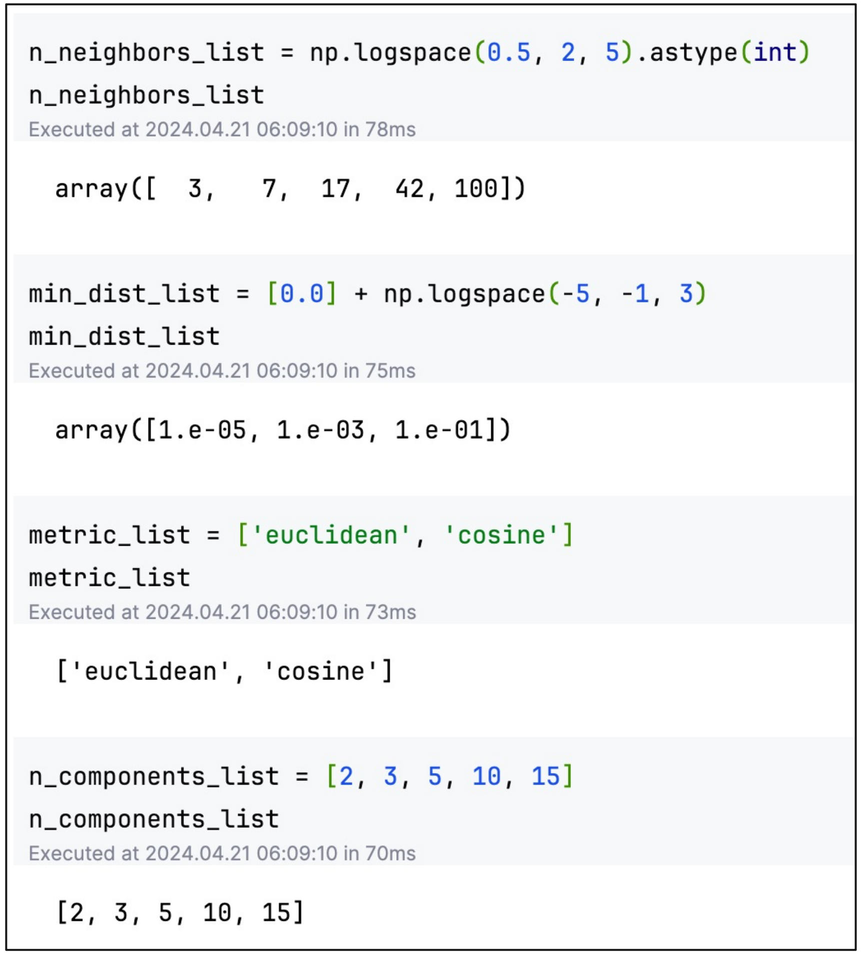
Clustering Pipeline Optimization: Hyperparameters

- The pipeline optimization refers to fine-tuning hyperparameters of UMAP, including several important parameters like **n_neighbors_list**, **min_dist_list**, **metric_list**, and **n_components_list**.

- We first tested a small proportion of the grid search space, to have an idea of the time consumption. It showed one hyperparameter combination takes around 3 - 5 minutes. We decided to explore a grid search space with 150 combinations, and it finally spent around 350 minutes.



	algo	eps	dbscan_min_samples	n_clusters_found	validity_index	hopkins_statistic	umap_n_neighbors	umap_mir
47	k_means	NaN	NaN	3	NaN	0.734961	7	
48	k_means	NaN	NaN	3	NaN	0.761623	7	
38	k_means	NaN	NaN	3	NaN	0.760266	7	
39	k_means	NaN	NaN	3	NaN	0.763695	7	
37	k_means	NaN	NaN	3	NaN	0.736636	7	
...	...	...	...	...	...	...	...	
110	dbscan	0.115614	6.0	3	-0.736907	0.453044	42	
85	dbscan	0.152404	6.0	3	-0.743036	0.424897	17	
100	dbscan	0.096328	6.0	2	-0.752965	0.449576	42	
25	dbscan	0.473131	6.0	13	-0.773474	0.332102	3	
45	dbscan	0.271902	6.0	2	-0.797766	0.367107	7	



```
n_neighbors_list = np.logspace(0.5, 2, 5).astype(int)
n_neighbors_list
Executed at 2024.04.21 06:09:10 in 78ms

 array([  3,   7,  17,  42, 100])
```

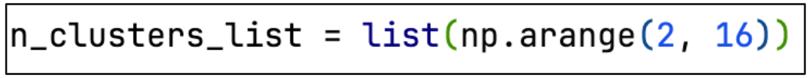
```
min_dist_list = [0.0] + np.logspace(-5, -1, 3)
min_dist_list
Executed at 2024.04.21 06:09:10 in 75ms

 array([1.e-05, 1.e-03, 1.e-01])
```

```
metric_list = ['euclidean', 'cosine']
metric_list
Executed at 2024.04.21 06:09:10 in 73ms

 ['euclidean', 'cosine']
```

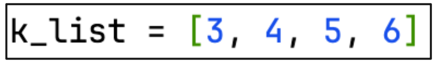
```
n_components_list = [2, 3, 5, 10, 15]
n_components_list
Executed at 2024.04.21 06:09:10 in 70ms

 [2, 3, 5, 10, 15]
```

Clustering Pipeline Optimization: Hyperparameters (continued)

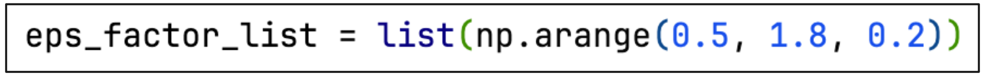
- Some hyperparameters of K-Means and DBSCAN were also tuned during the optimization process.
  - **n_clusters** of K-Means
  - **eps** of DBSCAN
  - **min_samples** of DBSCAN
  - **metric** of DBSCAN



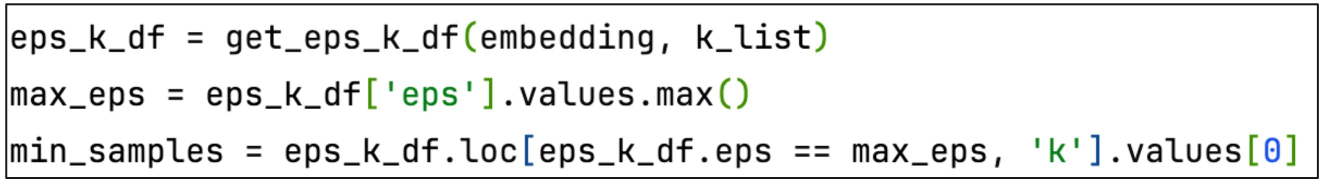
```
n_clusters_list = list(np.arange(2, 16))
```



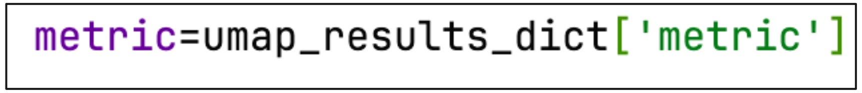
```
k_list = [3, 4, 5, 6]
```



```
eps_factor_list = list(np.arange(0.5, 1.8, 0.2))
```



```
eps_k_df = get_eps_k_df(embedding, k_list)
max_eps = eps_k_df['eps'].values.max()
min_samples = eps_k_df.loc[eps_k_df.eps == max_eps, 'k'].values[0]
```



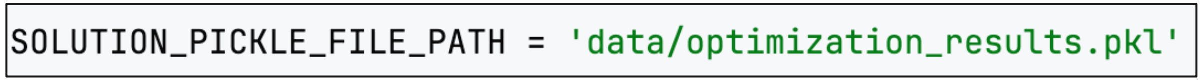
```
metric=umap_results_dict['metric']
```

Clustering Pipeline Optimization: Results

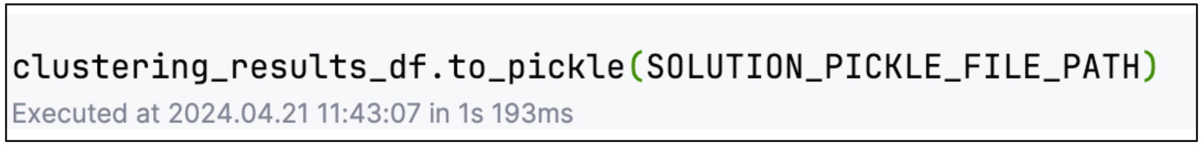
- Internal indices are used to select the best results during the pipeline optimization, includes:
  - **Silhouette Score** of K_Means
  - **Validity Index** of DBSCAN

- The result of pipeline optimization is a DataFrame of shape **(150, 19)**, including hyperparameter combinations, their resulting internal indices, UMAP embeddings, and fitted models.

- It was saved into an **optimization_results.pkl** file for the access of the following False Discovery Checking and Validation processes.



```
SOLUTION_PICKLE_FILE_PATH = 'data/optimization_results.pkl'
```



```
clustering_results_df.to_pickle(SOLUTION_PICKLE_FILE_PATH)
```

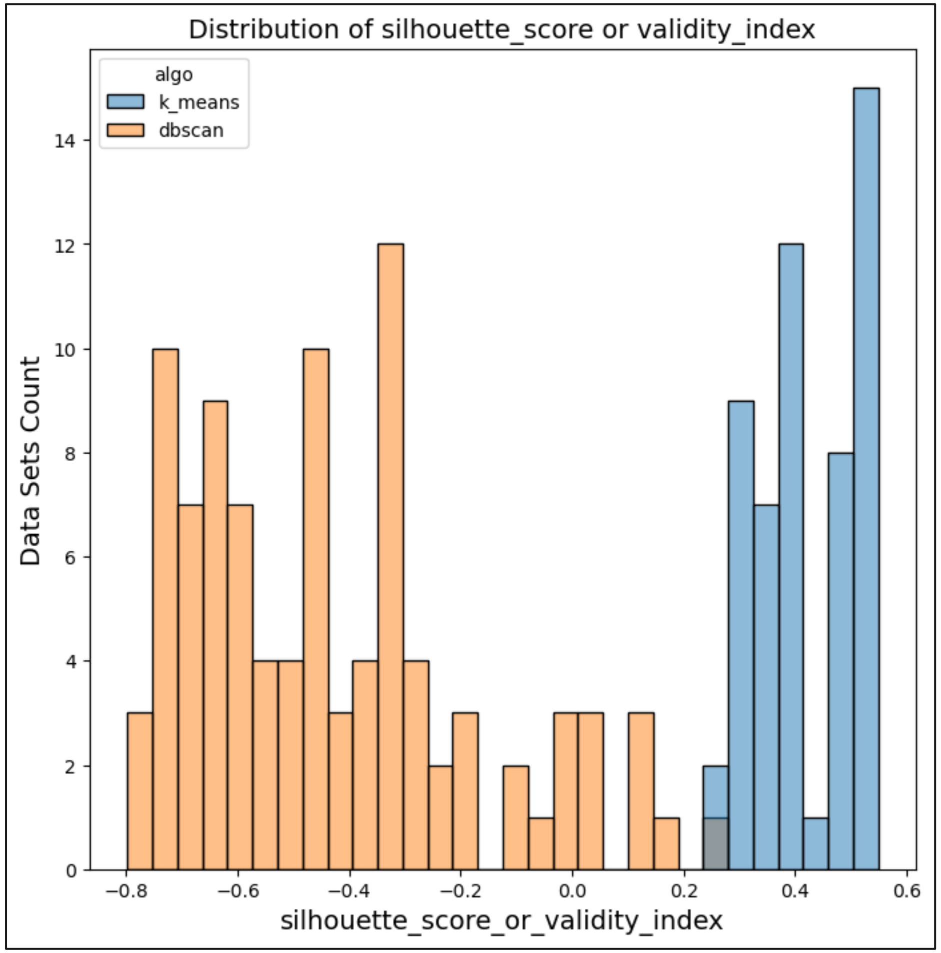
Executed at 2024.04.21 11:43:07 in 1s 193ms

Avoid False Discovery: Internal Indices Distribution

- As the optimization takes quite a long time, we put all results analysis and false discovery checking content in the **afd** notebook.

- Among the 150 clustering results, we put the **silhouette_score** of K_means and **validity_index** of DBSCAN together as the rank of results.
  - 1 means best clustering
  - 0 means overlapping clusters
  - -1 means worst clustering

- The distribution of **silhouette_score** or **validity_index** among the 150 results shows that **K_means performs better than DBSCAN** in general. And most DBSCAN have negative validity indices.



Avoid False Discovery: Latent Manifold Details

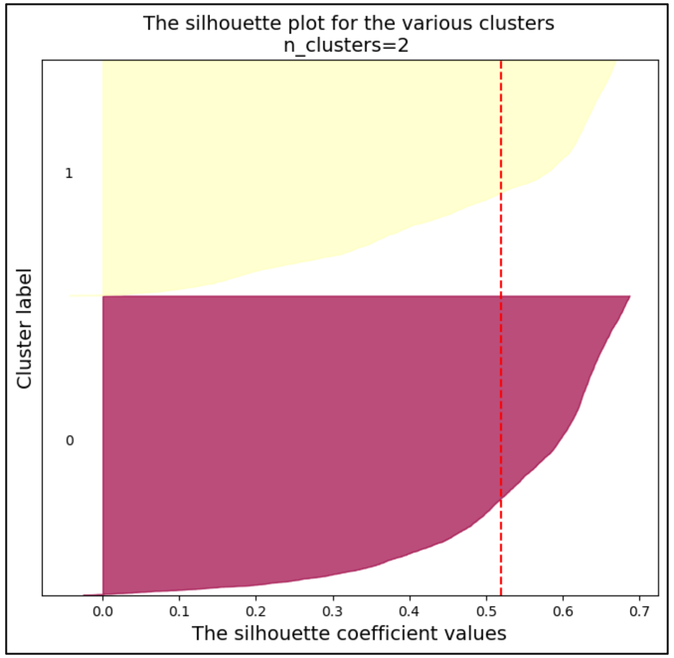
- The best result was chosen as the Latent Manifold and here is the hyperparameters and internal indices:

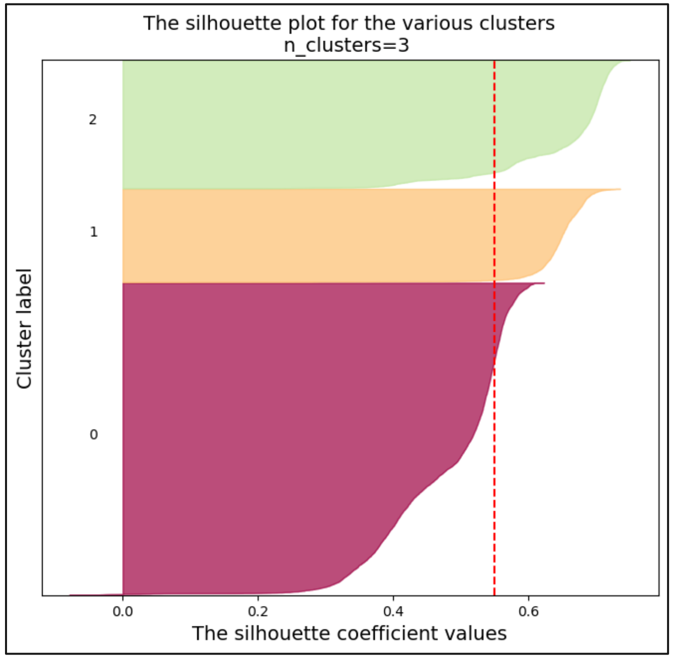
number of classes in data set	UMAP n_components	UMAP min_dist	UMAP n_neighbors	UMAP metric	trustworthiness	clustering algorithm	number of clusters found	validity index or silhouette score
4	5	0.001	7	cosine	0.823645	k_means	3	0.549104

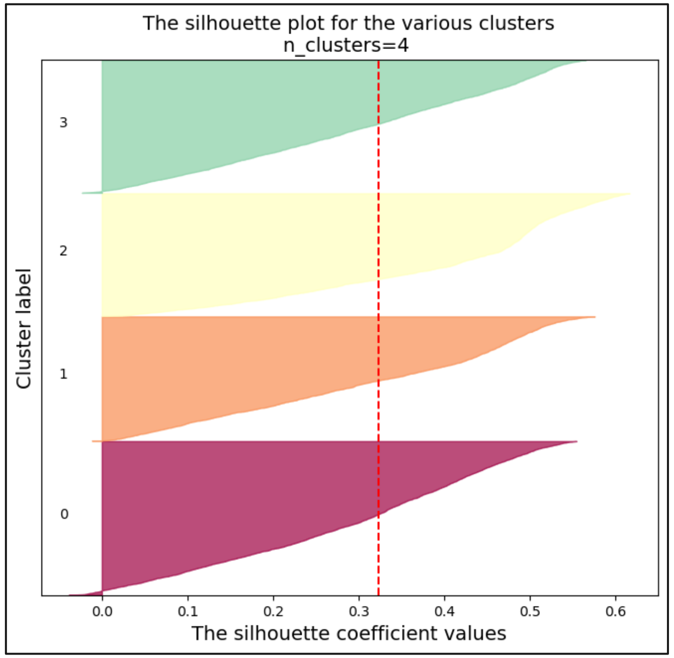
- It shows the highest validity_index or silhouette_score is **0.55**, which is not very high. We will discuss this in detail in the validation part.

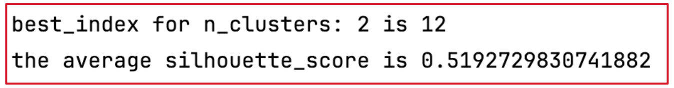
Avoid False Discovery: Silhouette Analysis

- Although we have chosen the latent manifold with the highest silhouette_score, we also did some exploration using the silhouette_score plots.
- Among the 150 clustering results, all ones using **k_means** algorithm have the following **n_clusters_found**: **2, 3, and 4**. For each of them, we select the best one result with the highest **silhouette_score** to analyze.

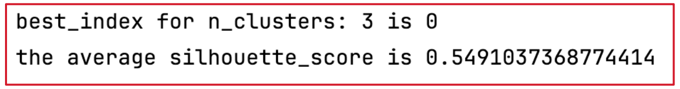




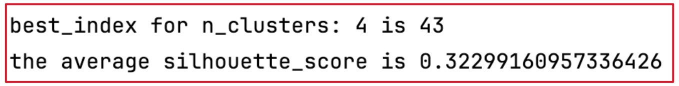




```
best_index for n_clusters: 2 is 12
the average silhouette_score is 0.5192729830741882
```



```
best_index for n_clusters: 3 is 0
the average silhouette_score is 0.5491037368774414
```



```
best_index for n_clusters: 4 is 43
the average silhouette_score is 0.32299160957336426
```

Avoid False Discovery: Silhouette Analysis (continued)

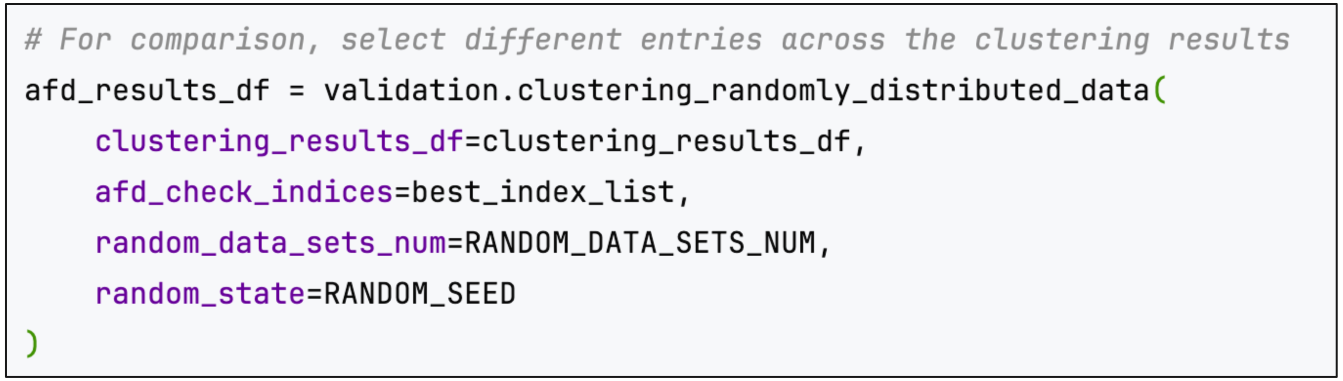
- The 3 silhouette_score plots show that although n_clusters=3 has the highest silhouette_score, n_clusters=2 and n_clusters=4 also have some meaningful results.
  - Width of the silhouette plot (y-axis) represents **the number of points in each cluster**. A wider silhouette indicates more points in the cluster.
  - Height of the silhouette plot (x-axis) represents **the silhouette score**. A silhouette that reaches closer to 1 indicates better clustering.
  - The red dashed line represents **the average silhouette score across all samples**. The closer the silhouette's height is to this line (or beyond), the better it is considered for the clustering.
- Although n_clusters=3 has the highest overall average silhouette score, indicating better average separation and cohesion, n_clusters=2 and n_clusters=4 also show that **the additional clusters' silhouettes are above the average line**. This means the existence of one extra cluster doesn't deteriorate the overall structure, and the additional clusters also hold statistically meaningful cohesion and separation.

Avoid False Discovery: Checking Sets

- Although we choose silhouette_score as the standard to rank clustering results, other clustering results may also be meaningful. Thus, in the False Discovery Checking part, we continue to check false discoveries for all the above 3 clustering results.

- For each clustering result, we use its UMAP embedding to generate randomly distributed data sets, and calculate their silhouette scores. **For a true discovery, its silhouette score should separate clearly from those of randomly distributed data sets.**

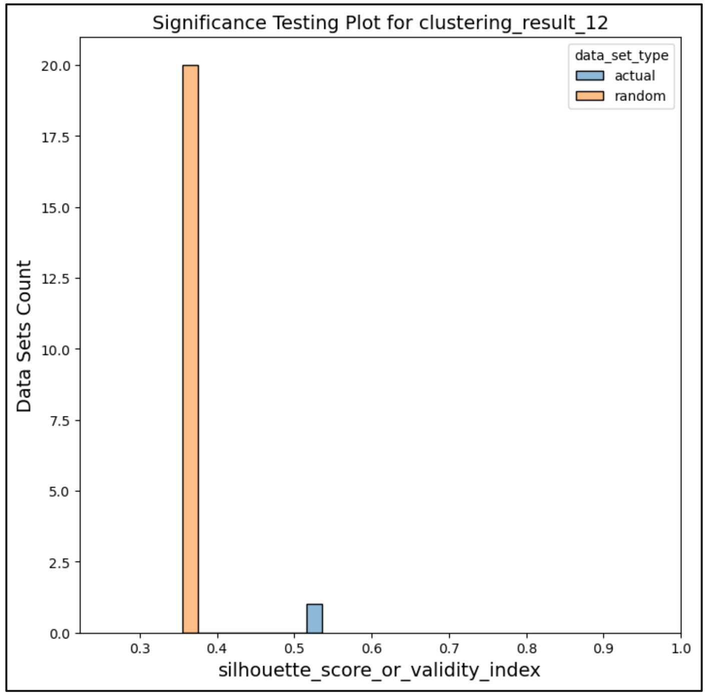
- The number of generated random sets (**random_data_sets_num**) can be declared and assigned to the AFD function. Also, we use one parameter **afd_check_indices** to determine which indices we are going to check. (here is the 3 results above)



```
# For comparison, select different entries across the clustering results
afd_results_df = validation.clustering_randomly_distributed_data(
    clustering_results_df=clustering_results_df,
    afd_check_indices=best_index_list,
    random_data_sets_num=RANDOM_DATA_SETS_NUM,
    random_state=RANDOM_SEED
)
```

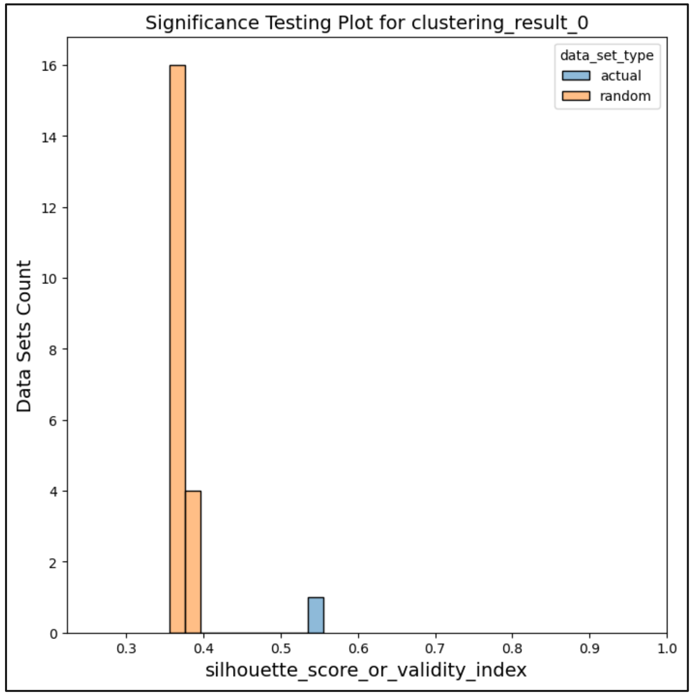
Avoid False Discovery: Checking Results

- The false discovery checking results are shown in the 3 plots below. It shows that **both n_clusters=2 and n_clusters=3 have a good separation with random data sets**, while n_cluters=4 has much overlapped with random sets.
- Considering the high silhouette score, **it seems n_clusters=3 is a good clustering result**.



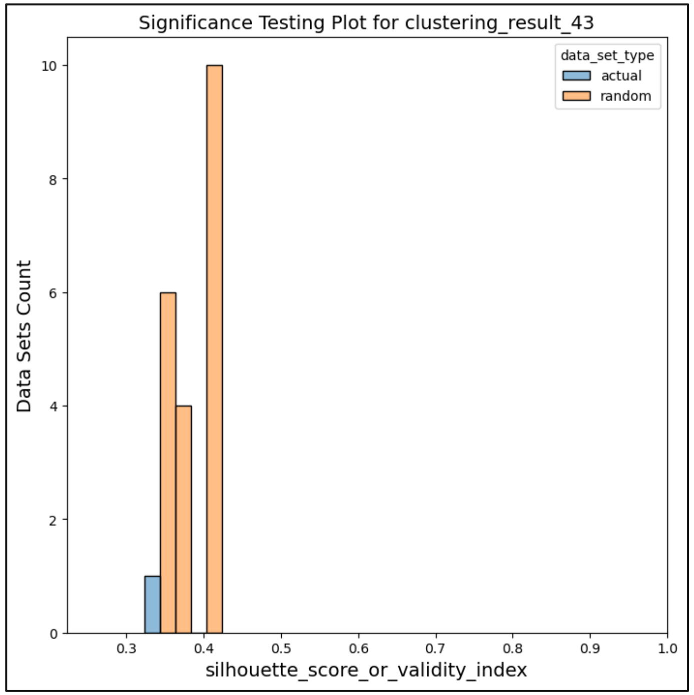
Significance Testing Plot for clustering_result_12

```
best_index for n_clusters: 2 is 12
the average silhouette_score is 0.5192729830741882
```



Significance Testing Plot for clustering_result_0

```
best_index for n_clusters: 3 is 0
the average silhouette_score is 0.5491037368774414
```

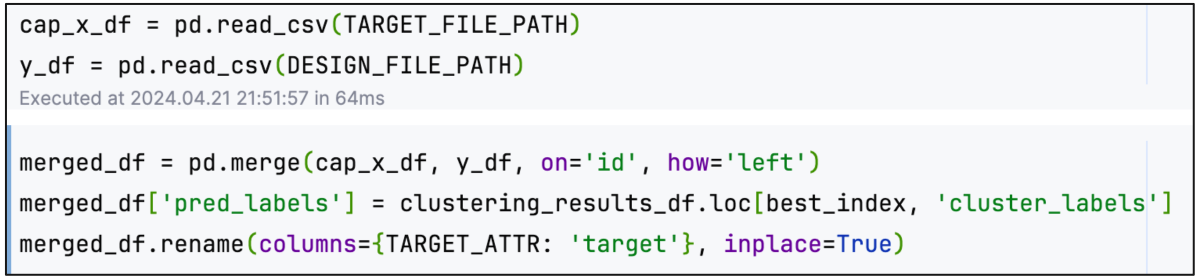


Significance Testing Plot for clustering_result_43

```
best_index for n_clusters: 4 is 43
the average silhouette_score is 0.32299160957336426
```

Clustering Validation: External Indices

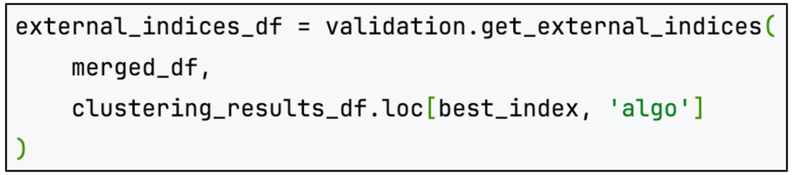
- In this part, we read back the pre-separated **target file** and **design file** in project phase 1, merge them back by **id**, add the **predicted label** from the best clustering result (the latent manifold), and calculate its external indices.
    - **rand_score**: [0, 1], 1 means perfect clustering
    - **adjusted_rand_score**: [-1, 1], 1 means perfect clustering
    - **fowlkes_mallows_score**: [0, 1], 1 means perfect clustering
    - **normalized_mutual_info_score**: [0, 1], 1 means perfect clustering
    - **jaccard_score**: scores for each label. [0, 1], 1 means perfect clustering
    - **f1_score**: scores for each label. [0, 1], 1 means perfect clustering
    - **purity_score**: [0, 1], 1 means perfect clustering
    - contingency_matrix



```
cap_x_df = pd.read_csv(TARGET_FILE_PATH)
y_df = pd.read_csv(DESIGN_FILE_PATH)
```

Executed at 2024.04.21 21:51:57 in 64ms

```
merged_df = pd.merge(cap_x_df, y_df, on='id', how='left')
merged_df['pred_labels'] = clustering_results_df.loc[best_index, 'cluster_labels']
merged_df.rename(columns={TARGET_ATTR: 'target'}, inplace=True)
```



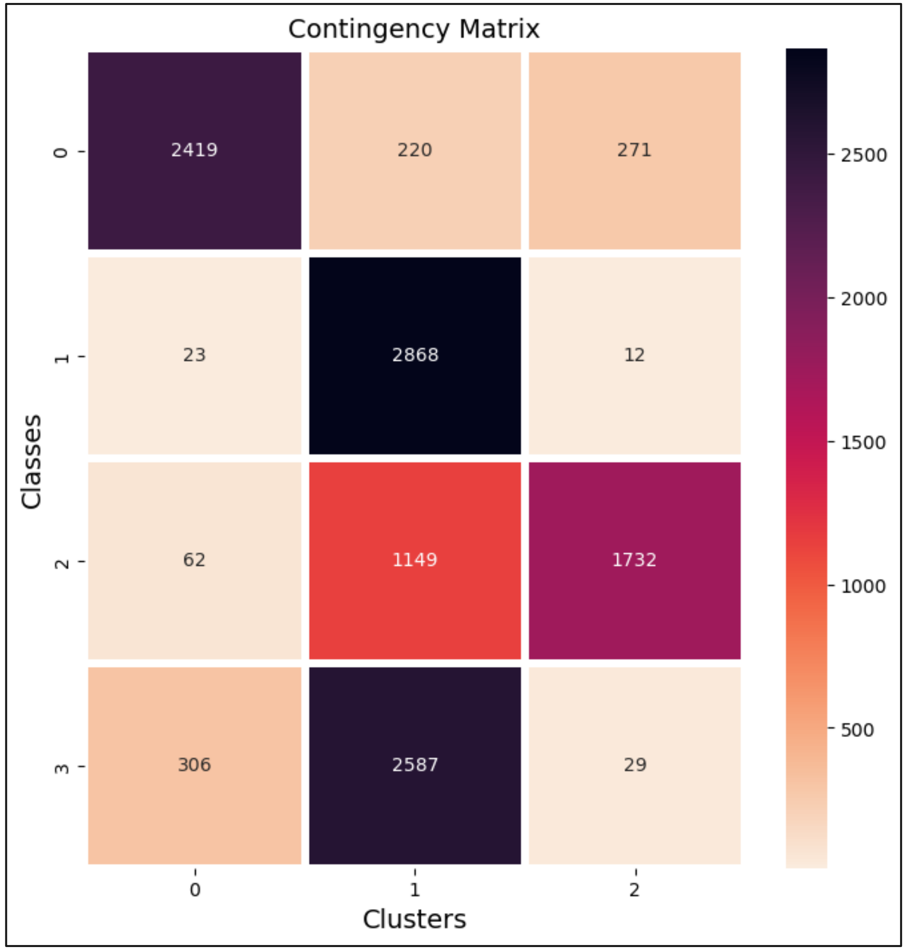
```
external_indices_df = validation.get_external_indices(
    merged_df,
    clustering_results_df.loc[best_index, 'algo']
)
```

Clustering Validation: External Indices Results

- The external indices results and the contingency matrix are generated in a data frame, they are also shown in the below table and the matrix plot.

rand_score	adjusted_rand_score	fowlkes_mallows_score	normalized_mutual_info_score
0.691157	0.335773	0.565941	0.436435

jaccard_score	f1_score	purity_score
[0.02312381753205802, 0.0024316109422492403, 0.010894394658232297, 0.0]	[0.04520238339839737, 0.0048514251061249234, 0.021553971840778726, 0.0]	0.601045



Discussion of Performance

- The number of clusters in our result (3) is different from the true number of classes (4). Specifically, the class 3 is not recognized and is mixed up with cluster 1.

- The external indices show some relatively good results, but not that good enough due to the different number of clusters with the true label numbers. Notice that jaccard_score and f1_score are label-based, which shows 0 score for the missing label.

- As the time limit, we can only perform current hyperparameter searching and get our current best result. For future research, maybe a broader search space, or different algorithms such as deep learnings can be applied to improve the performance.