

a6-mllab-ashwinravi

March 22, 2024

Ashwin Ravi

CSE-A

3122 21 5001 014

1 A6:K-means clustering algorithm

Github link:- <https://github.com/SolitudeAsh/Machine-Learning-Laboratory/tree/main/A6>

```
[1]: import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk("C:/Users/ashwi/Downloads/ML Lab/A6/
↳Input"):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

C:/Users/ashwi/Downloads/ML Lab/A6/Input\test.csv

C:/Users/ashwi/Downloads/ML Lab/A6/Input\train.csv

C:\Users\ashwi\AppData\Local\Temp\ipykernel_3160\419582161.py:2:

DeprecationWarning:

Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),

(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)

but was not found to be installed on your system.

If this would cause problems for you,

please provide us feedback at <https://github.com/pandas-dev/pandas/issues/54466>

```
import pandas as pd
```

```
[3]: import pandas as pd
import numpy as np

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

```

from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA

from sklearn.linear_model import Ridge
from sklearn.feature_selection import SelectFromModel

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score

```

Reading the Training and Testing Dataset

```

[5]: train_df=pd.read_csv("C:/Users/ashwi/Downloads/ML Lab/A6/Input/train.csv")
test_df=pd.read_csv("C:/Users/ashwi/Downloads/ML Lab/A6/Input/test.csv")

```

Display the Details about Training and Testing Dataset details

```

[6]: print("\n\nThe Size of The Training Dataset : ",train_df.shape)
print("\n\nThe Size of The Training Dataset : ",test_df.shape)

```

The Size of The Training Dataset : (7352, 563)

The Size of The Training Dataset : (2947, 563)

Printing The Training and Testing Dataset Examples

```

[7]: train_df

```

```

[7]:      tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
0          0.288585          -0.020294          -0.132905
1          0.278419          -0.016411          -0.123520
2          0.279653          -0.019467          -0.113462
3          0.279174          -0.026201          -0.123283
4          0.276629          -0.016570          -0.115362
...          ...          ...          ...
7347        0.299665          -0.057193          -0.181233
7348        0.273853          -0.007749          -0.147468
7349        0.273387          -0.017011          -0.045022
7350        0.289654          -0.018843          -0.158281
7351        0.351503          -0.012423          -0.203867

```

	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	\
0	-0.995279	-0.983111	-0.913526	-0.995112	
1	-0.998245	-0.975300	-0.960322	-0.998807	
2	-0.995380	-0.967187	-0.978944	-0.996520	
3	-0.996091	-0.983403	-0.990675	-0.997099	
4	-0.998139	-0.980817	-0.990482	-0.998321	
...	
7347	-0.195387	0.039905	0.077078	-0.282301	
7348	-0.235309	0.004816	0.059280	-0.322552	
7349	-0.218218	-0.103822	0.274533	-0.304515	
7350	-0.219139	-0.111412	0.268893	-0.310487	
7351	-0.269270	-0.087212	0.177404	-0.377404	

	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	\
0	-0.983185	-0.923527	-0.934724	...	
1	-0.974914	-0.957686	-0.943068	...	
2	-0.963668	-0.977469	-0.938692	...	
3	-0.982750	-0.989302	-0.938692	...	
4	-0.979672	-0.990441	-0.942469	...	
...	
7347	0.043616	0.060410	0.210795	...	
7348	-0.029456	0.080585	0.117440	...	
7349	-0.098913	0.332584	0.043999	...	
7350	-0.068200	0.319473	0.101702	...	
7351	-0.038678	0.229430	0.269013	...	

	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)	\
0	-0.710304	-0.112754	
1	-0.861499	0.053477	
2	-0.760104	-0.118559	
3	-0.482845	-0.036788	
4	-0.699205	0.123320	
...	
7347	-0.880324	-0.190437	
7348	-0.680744	0.064907	
7349	-0.304029	0.052806	
7350	-0.344314	-0.101360	
7351	-0.740738	-0.280088	

	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)	\
0	0.030400	-0.464761	
1	-0.007435	-0.732626	
2	0.177899	0.100699	
3	-0.012892	0.640011	
4	0.122542	0.693578	
...	
7347	0.829718	0.206972	

7348	0.875679	-0.879033
7349	-0.266724	0.864404
7350	0.700740	0.936674
7351	-0.007739	-0.056088

	angle(tBodyGyroJerkMean,gravityMean)	angle(X,gravityMean) \
0	-0.018446	-0.841247
1	0.703511	-0.844788
2	0.808529	-0.848933
3	-0.485366	-0.848649
4	-0.615971	-0.847865
...
7347	-0.425619	-0.791883
7348	0.400219	-0.771840
7349	0.701169	-0.779133
7350	-0.589479	-0.785181
7351	-0.616956	-0.783267

	angle(Y,gravityMean)	angle(Z,gravityMean)	subject	Activity
0	0.179941	-0.058627	1	STANDING
1	0.180289	-0.054317	1	STANDING
2	0.180637	-0.049118	1	STANDING
3	0.181935	-0.047663	1	STANDING
4	0.185151	-0.043892	1	STANDING
...
7347	0.238604	0.049819	30	WALKING_UPSTAIRS
7348	0.252676	0.050053	30	WALKING_UPSTAIRS
7349	0.249145	0.040811	30	WALKING_UPSTAIRS
7350	0.246432	0.025339	30	WALKING_UPSTAIRS
7351	0.246809	0.036695	30	WALKING_UPSTAIRS

[7352 rows x 563 columns]

[8]: test_df

[8]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z \
0	0.257178	-0.023285	-0.014654
1	0.286027	-0.013163	-0.119083
2	0.275485	-0.026050	-0.118152
3	0.270298	-0.032614	-0.117520
4	0.274833	-0.027848	-0.129527
...
2942	0.310155	-0.053391	-0.099109
2943	0.363385	-0.039214	-0.105915
2944	0.349966	0.030077	-0.115788
2945	0.237594	0.018467	-0.096499
2946	0.153627	-0.018437	-0.137018

	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	\
0	-0.938404	-0.920091	-0.667683	-0.952501	
1	-0.975415	-0.967458	-0.944958	-0.986799	
2	-0.993819	-0.969926	-0.962748	-0.994403	
3	-0.994743	-0.973268	-0.967091	-0.995274	
4	-0.993852	-0.967445	-0.978295	-0.994111	
...	
2942	-0.287866	-0.140589	-0.215088	-0.356083	
2943	-0.305388	0.028148	-0.196373	-0.373540	
2944	-0.329638	-0.042143	-0.250181	-0.388017	
2945	-0.323114	-0.229775	-0.207574	-0.392380	
2946	-0.330046	-0.195253	-0.164339	-0.430974	

	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	\
0	-0.925249	-0.674302	-0.894088	...	
1	-0.968401	-0.945823	-0.894088	...	
2	-0.970735	-0.963483	-0.939260	...	
3	-0.974471	-0.968897	-0.938610	...	
4	-0.965953	-0.977346	-0.938610	...	
...	
2942	-0.148775	-0.232057	0.185361	...	
2943	-0.030036	-0.270237	0.185361	...	
2944	-0.133257	-0.347029	0.007471	...	
2945	-0.279610	-0.289477	0.007471	...	
2946	-0.218295	-0.229933	-0.111527	...	

	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)	\
0	-0.705974	0.006462	
1	-0.594944	-0.083495	
2	-0.640736	-0.034956	
3	-0.736124	-0.017067	
4	-0.846595	-0.002223	
...	
2942	-0.750809	-0.337422	
2943	-0.700274	-0.736701	
2944	-0.467179	-0.181560	
2945	-0.617737	0.444558	
2946	-0.436940	0.598808	

	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)	\
0	0.162920	-0.825886	
1	0.017500	-0.434375	
2	0.202302	0.064103	
3	0.154438	0.340134	
4	-0.040046	0.736715	
...	

2942	0.346295	0.884904
2943	-0.372889	-0.657421
2944	0.088574	0.696663
2945	-0.819188	0.929294
2946	-0.287951	0.876030

	angle(tBodyGyroJerkMean,gravityMean)	angle(X,gravityMean)	\
0	0.271151	-0.720009	
1	0.920593	-0.698091	
2	0.145068	-0.702771	
3	0.296407	-0.698954	
4	-0.118545	-0.692245	
...	
2942	-0.698885	-0.651732	
2943	0.322549	-0.655181	
2944	0.363139	-0.655357	
2945	-0.008398	-0.659719	
2946	-0.024965	-0.660080	

	angle(Y,gravityMean)	angle(Z,gravityMean)	subject	Activity
0	0.276801	-0.057978	2	STANDING
1	0.281343	-0.083898	2	STANDING
2	0.280083	-0.079346	2	STANDING
3	0.284114	-0.077108	2	STANDING
4	0.290722	-0.073857	2	STANDING
...
2942	0.274627	0.184784	24	WALKING_UPSTAIRS
2943	0.273578	0.182412	24	WALKING_UPSTAIRS
2944	0.274479	0.181184	24	WALKING_UPSTAIRS
2945	0.264782	0.187563	24	WALKING_UPSTAIRS
2946	0.263936	0.188103	24	WALKING_UPSTAIRS

[2947 rows x 563 columns]

Data Preprocessing (Handling Missing values)

```
[9]: print("The Missing Values in The Training Dataset\n\n",train_df.isnull().sum())
```

The Missing Values in The Training Dataset

tBodyAcc-mean()-X	0
tBodyAcc-mean()-Y	0
tBodyAcc-mean()-Z	0
tBodyAcc-std()-X	0
tBodyAcc-std()-Y	0
..	
angle(X,gravityMean)	0
angle(Y,gravityMean)	0

```
angle(Z,gravityMean)    0
subject                0
Activity               0
Length: 563, dtype: int64
```

```
[10]: print("The Missing Values in The Testing Dataset\n\n",test_df.isnull().sum())
```

The Missing Values in The Testing Dataset

```
tBodyAcc-mean()-X      0
tBodyAcc-mean()-Y      0
tBodyAcc-mean()-Z      0
tBodyAcc-std()-X       0
tBodyAcc-std()-Y       0
..
angle(X,gravityMean)   0
angle(Y,gravityMean)   0
angle(Z,gravityMean)   0
subject               0
Activity              0
Length: 563, dtype: int64
```

Display The Features in Dataset

```
[11]: print(train_df.columns)
```

```
Index(['tBodyAcc-mean()-X', 'tBodyAcc-mean()-Y', 'tBodyAcc-mean()-Z',
      'tBodyAcc-std()-X', 'tBodyAcc-std()-Y', 'tBodyAcc-std()-Z',
      'tBodyAcc-mad()-X', 'tBodyAcc-mad()-Y', 'tBodyAcc-mad()-Z',
      'tBodyAcc-max()-X',
      ...,
      'fBodyBodyGyroJerkMag-kurtosis()', 'angle(tBodyAccMean,gravity)',
      'angle(tBodyAccJerkMean,gravityMean)',
      'angle(tBodyGyroMean,gravityMean)',
      'angle(tBodyGyroJerkMean,gravityMean)', 'angle(X,gravityMean)',
      'angle(Y,gravityMean)', 'angle(Z,gravityMean)', 'subject', 'Activity'],
      dtype='object', length=563)
```

```
[12]: x=train_df.drop(columns={"Activity"})
      y=train_df["Activity"]
```

Feature Engineering Techniques

- 1) Select Best K (Filter method)
- 2) Ridge Regression (Embedded Method)
- 3) PCA

```
[13]: test = SelectKBest(score_func=f_classif, k=5)
fit = test.fit(x, y)
np.set_printoptions(precision=10)

features = fit.transform(x)

selected_indices = fit.get_support(indices=True)

selected_feature_names = x.columns[selected_indices]

print("Selected feature names : \n")
print(selected_feature_names)
```

Selected feature names :

```
Index(['tGravityAcc-mean()-X', 'tGravityAcc-max()-X', 'tGravityAcc-min()-X',
      'fBodyAccJerk-entropy()-X', 'fBodyAccJerk-entropy()-Y'],
      dtype='object')
```

```
[14]: train_df1 = train_df[selected_feature_names].copy()
test_df1 = test_df[selected_feature_names].copy()
```

```
[15]: print("\nAfter Feature Selection The Shape of The Training Dataset : \n",
      ↪train_df1.shape)
print("After Feature Selection The Shape of The Testing Dataset : \n",
      ↪test_df1.shape)
```

After Feature Selection The Shape of The Training Dataset : (7352, 5)

After Feature Selection The Shape of The Testing Dataset : (2947, 5)

```
[16]: n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=42)
kmeans.fit(train_df1)
train_clusters = kmeans.predict(train_df1)
test_clusters = kmeans.predict(test_df1)
cluster_centroids = kmeans.cluster_centers_
```

```
[18]: silhouette_avg11 = silhouette_score(train_df1, train_clusters)
silhouette_avg12 = silhouette_score(test_df1, test_clusters)
print(f"Silhouette Score ( Training dataset ) : {silhouette_avg11}")
print(f"Silhouette Score ( Testing dataset ) : {silhouette_avg12}")
```

```
-----
MemoryError                                Traceback (most recent call last)
Cell In[18], line 1
----> 1 silhouette_avg11 = silhouette_score(train_df1, train_clusters)
```

```

2 silhouette_avg12 = silhouette_score(test_df1, test_clusters)
3 print(f"Silhouette Score ( Training dataset ) : {silhouette_avg11}")

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_validation.py:213, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    207 try:
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213         return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we
↪replace
    217     # the name of the estimator by the name of the function in the error
    218     # message to avoid confusion.
    219     msg = re.sub(
    220         r"parameter of \w+ must be",
    221         f"parameter of {func.__qualname__} must be",
    222         str(e),
    223     )

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\cluster.py:140, in silhouette_score(X, labels, metric, sample_size, random_state, **kwargs)
    138 else:
    139     X, labels = X[indices], labels[indices]
--> 140 return np.mean(silhouette_samples(X, labels, metric=metric, **kwargs))

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_validation.py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\cluster.py:292, in silhouette_samples(X, labels, metric, **kwargs)
    288 kwds["metric"] = metric
    289 reduce_func = functools.partial(
    290     _silhouette_reduce, labels=labels, label_freqs=label_freqs
    291 )

```

```

--> 292 results =
↳ zip(*pairwise_distances_chunked(X, reduce_func=reduce_func, **kws))
    293 intra_clust_dists, inter_clust_dists = results
    294 intra_clust_dists = np.concatenate(intra_clust_dists)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↳ py:2144, in pairwise_distances_chunked(X, Y, reduce_func, metric, n_jobs,
↳ working_memory, **kws)
    2142 else:
    2143     X_chunk = X[s1]
-> 2144 D_chunk = pairwise_distances(X_chunk, Y, metric=metric, n_jobs=n_jobs,
↳ **kws)
    2145 if (X is Y or Y is None) and PAIRWISE_DISTANCE_FUNCTIONS.get(
    2146     metric, None
    2147 ) is euclidean_distances:
    2148     # zeroing diagonal, taking care of aliases of "euclidean",
    2149     # i.e. "l2"
    2150     D_chunk.flat[s1.start :: _num_samples(X) + 1] = 0

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
↳ py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↳ py:2331, in pairwise_distances(X, Y, metric, n_jobs, force_all_finite, **kws)
    2328     return distance.squareform(distance.pdist(X, metric=metric,
↳ **kws))
    2329     func = partial(distance.cdist, metric=metric, **kws)
-> 2331 return _parallel_pairwise(X, Y, func, n_jobs, **kws)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↳ py:1871, in _parallel_pairwise(X, Y, func, n_jobs, **kws)
    1868 X, Y, dtype = _return_float_dtype(X, Y)
    1870 if effective_n_jobs(n_jobs) == 1:
-> 1871     return func(X, Y, **kws)
    1873 # enforce a threading backend to prevent data communication overhead
    1874 fd = delayed(_dist_wrapper)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
↳ py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)

```

```

184 global_skip_validation = get_config()["skip_parameter_validation"]
185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
188 func_sig = signature(func)
190 # Map *args/**kwargs to the function signature

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↪py:347, in euclidean_distances(X, Y, Y_norm_squared, squared, X_norm_squared)
341     if Y_norm_squared.shape != (1, Y.shape[0]):
342         raise ValueError(
343             f"Incompatible dimensions for Y of shape {Y.shape} and "
344             f"Y_norm_squared of shape {original_shape}."
345         )
--> 347 return
↪_euclidean_distances(X, Y, X_norm_squared, Y_norm_squared, squared)

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↪py:382, in _euclidean_distances(X, Y, X_norm_squared, Y_norm_squared, squared
379     distances = _euclidean_distances_upcast(X, XX, Y, YY)
380 else:
381     # if dtype is already float64, no need to chunk and upcast
--> 382     distances = -2 * safe_sparse_dot(X, Y.T, dense_output=True)
383     distances += XX
384     distances += YY

```

MemoryError: Unable to allocate 412. MiB for an array with shape (7352, 7352) and data type float64

```

[47]: from sklearn.metrics import silhouette_score

# Define batch size for silhouette score computation
batch_size = 1000

# Compute silhouette score for training dataset
train_num_batches = len(train_df1) // batch_size + 1
silhouette_scores_train = []

for i in range(train_num_batches):
    start_idx = i * batch_size
    end_idx = min((i + 1) * batch_size, len(train_df1))
    train_batch = train_df1[start_idx:end_idx]
    train_clusters_batch = train_clusters[start_idx:end_idx]
    silhouette_avg_train_11 = silhouette_score(train_batch,
↪train_clusters_batch)
    silhouette_scores_train.append(silhouette_avg_train_11)

```

```

# Compute average silhouette score for training dataset
silhouette_avg_train_11 = sum(silhouette_scores_train) /
    len(silhouette_scores_train)

# Compute silhouette score for testing dataset
test_num_batches = len(test_df1) // batch_size + 1
silhouette_scores_test = []

for i in range(test_num_batches):
    start_idx = i * batch_size
    end_idx = min((i + 1) * batch_size, len(test_df1))
    test_batch = test_df1[start_idx:end_idx]
    test_clusters_batch = test_clusters[start_idx:end_idx]
    silhouette_avg_test_11 = silhouette_score(test_batch, test_clusters_batch)
    silhouette_scores_test.append(silhouette_avg_test_11)

# Compute average silhouette score for testing dataset
silhouette_avg_test_11 = sum(silhouette_scores_test) /
    len(silhouette_scores_test)

print(f"Silhouette Score (Training dataset): {silhouette_avg_train_11}")
print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_11}")

```

Silhouette Score (Training dataset): 0.3397089651970844

Silhouette Score (Testing dataset): 0.3529194692224184

```

[20]: plt.figure(figsize=(4, 4))

for cluster in range(n_clusters):
    cluster_data = train_df1[train_clusters == cluster]

    plt.scatter(cluster_data['tGravityAcc-mean()-X'],
        cluster_data['tGravityAcc-max()-X'], label=f'Cluster {cluster}')

plt.scatter(cluster_centroids[:, 0], cluster_centroids[:, 1], marker='x',
    color='black', s=100, label='Centroids')

plt.title('Clustering Results')
plt.xlabel('GravityAcc-mean()-X')
plt.ylabel('tGravityAcc-max()-X')
plt.legend()
plt.grid(True)
plt.show()

```



2) Ridge Regression (Embedded Method)

```
[21]: label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

ridge = Ridge(alpha=1.0)
select_from_model = SelectFromModel(ridge, max_features=8)
select_from_model.fit(x, y_encoded)
features_selected = select_from_model.transform(x)
selected_indices = select_from_model.get_support(indices=True)
selected_feature_names = x.columns[selected_indices]
print("Selected feature names : \n")
print(selected_feature_names)
```

Selected feature names :

```
Index(['tBodyAcc-std()-X', 'tBodyAcc-sma()', 'tGravityAcc-std()-X',
      'tBodyGyroJerk-std()-Z', 'tBodyGyroJerk-energy()-Z', 'fBodyAcc-std()-X',
      'fBodyGyro-bandsEnergy()-9,16.2', 'fBodyGyro-bandsEnergy()-17,32.2'],
      dtype='object')
```

```
[22]: train_df2 = train_df[selected_feature_names].copy()
test_df2 = test_df[selected_feature_names].copy()
```

```
[23]: n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=42)
kmeans.fit(train_df2)
train_clusters = kmeans.predict(train_df2)
test_clusters = kmeans.predict(test_df2)
cluster_centroids = kmeans.cluster_centers_
```

```
[24]: silhouette_avg21 = silhouette_score(train_df2, train_clusters)
silhouette_avg22 = silhouette_score(test_df2, test_clusters)
print(f"Silhouette Score ( Training dataset ) : {silhouette_avg21}")
print(f"Silhouette Score ( Testing dataset ) : {silhouette_avg22}")
```

```
-----
MemoryError                                Traceback (most recent call last)
```

```
Cell In[24], line 1
```

```
----> 1 silhouette_avg21 = silhouette_score(train_df2, train_clusters)
      2 silhouette_avg22 = silhouette_score(test_df2, test_clusters)
      3 print(f"Silhouette Score ( Training dataset ) : {silhouette_avg21}")
```

```
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
```

```
  <10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_validation.py>
```

```
  <py:213, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)>
```

```
    207 try:
```

```
    208     with config_context(
```

```
    209         skip_parameter_validation=(
```

```
    210             prefer_skip_nested_validation or global_skip_validation
```

```
    211         )
```

```
    212     ):
--> 213         return func(*args, **kwargs)
```

```
    214 except InvalidParameterError as e:
```

```
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we
```

```
  <replace>
```

```
    217     # the name of the estimator by the name of the function in the error
    218     # message to avoid confusion.
```

```
    219     msg = re.sub(
```

```
    220         r"parameter of \w+ must be",
```

```
    221         f"parameter of {func.__qualname__} must be",
```

```
    222         str(e),
```

```
    223     )
```

```
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
```

```
  <10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\cluster.py>
```

```
  <py:140, in silhouette_score(X, labels, metric, sample_size, random_state, **kwds)>
```

```
  <py:140, in silhouette_score(X, labels, metric, sample_size, random_state, **kwds)>
```

```
    138     else:
```

```
    139         X, labels = X[indices], labels[indices]
```

```
--> 140 return np.mean(silhouette_samples(X, labels, metric=metric, **kwds))
```

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
↳py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

```

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\clusterin
↳py:292, in silhouette_samples(X, labels, metric, **kws)
    288 kws["metric"] = metric
    289 reduce_func = functools.partial(
    290     _silhouette_reduce, labels=labels, label_freqs=label_freqs
    291 )
--> 292 results =
↳zip(*pairwise_distances_chunked(X, reduce_func=reduce_func, **kws))
    293 intra_clust_dists, inter_clust_dists = results
    294 intra_clust_dists = np.concatenate(intra_clust_dists)

```

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↳py:2144, in pairwise_distances_chunked(X, Y, reduce_func, metric, n_jobs,
↳working_memory, **kws)
    2142 else:
    2143     X_chunk = X[s1]
-> 2144 D_chunk = pairwise_distances(X_chunk, Y, metric=metric, n_jobs=n_jobs,
↳**kws)
    2145 if (X is Y or Y is None) and PAIRWISE_DISTANCE_FUNCTIONS.get(
    2146     metric, None
    2147 ) is euclidean_distances:
    2148     # zeroing diagonal, taking care of aliases of "euclidean",
    2149     # i.e. "l2"
    2150     D_chunk.flat[s1.start :: _num_samples(X) + 1] = 0

```

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
↳py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

```

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
↳py:2331, in pairwise_distances(X, Y, metric, n_jobs, force_all_finite, **kws)

```

```

2328         return distance.squareform(distance.pdist(X, metric=metric,
↳ **kws))
2329     func = partial(distance.cdist, metric=metric, **kws)
-> 2331 return _parallel_pairwise(X, Y, func, n_jobs, **kws)

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
py:1871, in _parallel_pairwise(X, Y, func, n_jobs, **kws)
1868 X, Y, dtype = _return_float_dtype(X, Y)
1870 if effective_n_jobs(n_jobs) == 1:
-> 1871     return func(X, Y, **kws)
1873 # enforce a threading backend to prevent data communication overhead
1874 fd = delayed(_dist_wrapper)

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
184 global_skip_validation = get_config()["skip_parameter_validation"]
185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
188 func_sig = signature(func)
190 # Map *args/**kwargs to the function signature

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
py:347, in euclidean_distances(X, Y, Y_norm_squared, squared, X_norm_squared)
341     if Y_norm_squared.shape != (1, Y.shape[0]):
342         raise ValueError(
343             f"Incompatible dimensions for Y of shape {Y.shape} and "
344             f"Y_norm_squared of shape {original_shape}."
345         )
--> 347 return
↳ _euclidean_distances(X, Y, X_norm_squared, Y_norm_squared, squared)

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↳ 10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise
py:382, in _euclidean_distances(X, Y, X_norm_squared, Y_norm_squared, squared)
379     distances = _euclidean_distances_upcast(X, XX, Y, YY)
380 else:
381     # if dtype is already float64, no need to chunk and upcast
--> 382     distances = -2 * safe_sparse_dot(X, Y.T, dense_output=True)
383     distances += XX
384     distances += YY

```

```

MemoryError: Unable to allocate 412. MiB for an array with shape (7352, 7352)
↳ and data type float64

```

```
[40]: from sklearn.metrics import silhouette_score

# Define batch size for silhouette score computation
batch_size = 1000

# Compute silhouette score for training dataset
train_num_batches = len(train_df2) // batch_size + 1
silhouette_scores_train = []

for i in range(train_num_batches):
    start_idx = i * batch_size
    end_idx = min((i + 1) * batch_size, len(train_df2))
    train_batch = train_df2[start_idx:end_idx]
    train_clusters_batch = train_clusters[start_idx:end_idx]
    silhouette_avg_train_21 = silhouette_score(train_batch,
    ↪train_clusters_batch)
    silhouette_scores_train.append(silhouette_avg_train_21)

# Compute average silhouette score for training dataset
silhouette_avg_train_21 = sum(silhouette_scores_train) /
    ↪len(silhouette_scores_train)

# Compute silhouette score for testing dataset
test_num_batches = len(test_df2) // batch_size + 1
silhouette_scores_test = []

for i in range(test_num_batches):
    start_idx = i * batch_size
    end_idx = min((i + 1) * batch_size, len(test_df2))
    test_batch = test_df2[start_idx:end_idx]
    test_clusters_batch = test_clusters[start_idx:end_idx]
    silhouette_avg_test_21 = silhouette_score(test_batch, test_clusters_batch)
    silhouette_scores_test.append(silhouette_avg_test_21)

# Compute average silhouette score for testing dataset
silhouette_avg_test_21 = sum(silhouette_scores_test) /
    ↪len(silhouette_scores_test)

print(f"Silhouette Score (Training dataset): {silhouette_avg_train_21}")
print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_21}")
```

Silhouette Score (Training dataset): 0.6952144230232754
 Silhouette Score (Testing dataset): 0.7131016420643701

```
[26]: plt.figure(figsize=(4, 4))

for cluster in range(n_clusters):
```

```

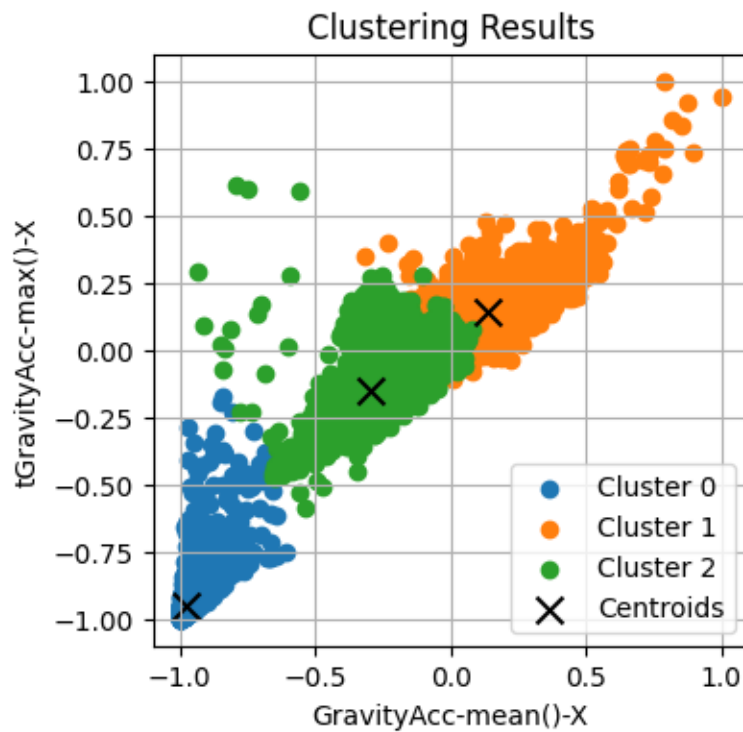
cluster_data = train_df2[train_clusters == cluster]

plt.scatter(cluster_data['tBodyAcc-std()-X'],
            cluster_data['tBodyAcc-sma()'], label=f'Cluster {cluster}')

plt.scatter(cluster_centroids[:, 0], cluster_centroids[:, 1], marker='x',
            color='black', s=100, label='Centroids')

plt.title('Clustering Results')
plt.xlabel('GravityAcc-mean()-X')
plt.ylabel('tGravityAcc-max()-X')
plt.legend()
plt.grid(True)
plt.show()

```



3) PCA

```

[27]: pca = PCA(n_components=11)

pca.fit(x)

selected_feature_indices = pca.components_

```

```
selected_feature_names = [x.columns[i] for i in
    ↪range(len(selected_feature_indices))]
```

```
print("Selected feature names:")
print(selected_feature_names)
```

Selected feature names:

```
['tBodyAcc-mean()-X', 'tBodyAcc-mean()-Y', 'tBodyAcc-mean()-Z', 'tBodyAcc-
std()-X', 'tBodyAcc-std()-Y', 'tBodyAcc-std()-Z', 'tBodyAcc-mad()-X', 'tBodyAcc-
mad()-Y', 'tBodyAcc-mad()-Z', 'tBodyAcc-max()-X', 'tBodyAcc-max()-Y']
```

```
[28]: train_df3 = train_df[selected_feature_names].copy()
test_df3 = test_df[selected_feature_names].copy()
```

```
[29]: n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=42)
kmeans.fit(train_df3)
train_clusters = kmeans.predict(train_df3)
test_clusters = kmeans.predict(test_df3)
cluster_centroids = kmeans.cluster_centers_
```

```
[30]: silhouette_avg31 = silhouette_score(train_df3, train_clusters)
silhouette_avg32 = silhouette_score(test_df3, test_clusters)
print(f"Silhouette Score ( Training dataset ) : {silhouette_avg31}")
print(f"Silhouette Score ( Testing dataset ) : {silhouette_avg32}")
```

```
-----
MemoryError                                Traceback (most recent call last)
Cell In[30], line 1
```

```
----> 1 silhouette_avg31 = silhouette_score(train_df3, train_clusters)
      2 silhouette_avg32 = silhouette_score(test_df3, test_clusters)
      3 print(f"Silhouette Score ( Training dataset ) : {silhouette_avg31}")
```

```
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
```

```
↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
```

```
↪py:213, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs):
```

```
    207 try:
```

```
    208     with config_context(
```

```
    209         skip_parameter_validation=(
```

```
    210             prefer_skip_nested_validation or global_skip_validation
```

```
    211         )
```

```
    212     ):
```

```
--> 213         return func(*args, **kwargs)
```

```
    214 except InvalidParameterError as e:
```

```
    215     # When the function is just a wrapper around an estimator, we allow
```

```
    216     # the function to delegate validation to the estimator, but we
```

```
↪replace
```

```

217     # the name of the estimator by the name of the function in the error
218     # message to avoid confusion.
219     msg = re.sub(
220         r"parameter of \w+ must be",
221         f"parameter of {func.__qualname__} must be",
222         str(e),
223     )

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\cluster
↪py:140, in silhouette_score(X, labels, metric, sample_size, random_state,
↪**kwargs)
    138     else:
    139         X, labels = X[indices], labels[indices]
--> 140 return np.mean(silhouette_samples(X, labels, metric=metric, **kwargs))

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_val
↪py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\cluster
↪py:292, in silhouette_samples(X, labels, metric, **kwargs)
    288 kwargs["metric"] = metric
    289 reduce_func = functools.partial(
    290     _silhouette_reduce, labels=labels, label_freqs=label_freqs
    291 )
--> 292 results =
↪zip(*pairwise_distances_chunked(X, reduce_func=reduce_func, **kwargs))
    293 intra_clust_dists, inter_clust_dists = results
    294 intra_clust_dists = np.concatenate(intra_clust_dists)

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.

```

↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwis
↪py:2144, in pairwise_distances_chunked(X, Y, reduce_func, metric, n_jobs,
↪working_memory, **kwargs)
    2142 else:
    2143     X_chunk = X[s1]
-> 2144 D_chunk = pairwise_distances(X_chunk, Y, metric=metric, n_jobs=n_jobs,
↪**kwargs)
    2145 if (X is Y or Y is None) and PAIRWISE_DISTANCE_FUNCTIONS.get(
    2146     metric, None
    2147 ) is euclidean_distances:
    2148     # zeroing diagonal, taking care of aliases of "euclidean",

```

```

2149     # i.e. "l2"
2150     D_chunk.flat[sl.start :: _num_samples(X) + 1] = 0

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_validation.py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise_distances.py:2331, in pairwise_distances(X, Y, metric, n_jobs, force_all_finite, **kwargs)
    2328     return distance.squareform(distance.pdist(X, metric=metric,
↳**kwargs))
    2329     func = partial(distance.cdist, metric=metric, **kwargs)
-> 2331 return _parallel_pairwise(X, Y, func, n_jobs, **kwargs)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise_distances.py:1871, in _parallel_pairwise(X, Y, func, n_jobs, **kwargs)
    1868 X, Y, dtype = _return_float_dtype(X, Y)
    1870 if effective_n_jobs(n_jobs) == 1:
-> 1871     return func(X, Y, **kwargs)
    1873 # enforce a threading backend to prevent data communication overhead
    1874 fd = delayed(_dist_wrapper)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\utils\_param_validation.py:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwise_distances.py:347, in euclidean_distances(X, Y, Y_norm_squared, squared, X_norm_squared)
    341     if Y_norm_squared.shape != (1, Y.shape[0]):
    342         raise ValueError(
    343             f"Incompatible dimensions for Y of shape {Y.shape} and "
    344             f"Y_norm_squared of shape {original_shape}."
    345         )
--> 347 return _
↳_euclidean_distances(X, Y, X_norm_squared, Y_norm_squared, squared)

```

```

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
  ↳10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\metrics\pairwis
  ↳py:382, in _euclidean_distances(X, Y, X_norm_squared, Y_norm_squared, squared
    379     distances = _euclidean_distances_upcast(X, XX, Y, YY)
    380 else:
    381     # if dtype is already float64, no need to chunk and upcast
--> 382     distances = -2 * safe_sparse_dot(X, Y.T, dense_output=True)
    383     distances += XX
    384     distances += YY

MemoryError: Unable to allocate 412. MiB for an array with shape (7352, 7352)
  ↳and data type float64

```

```

[36]: from sklearn.metrics import silhouette_score

# Define batch size for silhouette score computation
batch_size = 1000

# Compute silhouette score for training dataset
train_num_batches = len(train_df3) // batch_size + 1
silhouette_scores_train = []

for i in range(train_num_batches):
    start_idx = i * batch_size
    end_idx = min((i + 1) * batch_size, len(train_df3))
    train_batch = train_df3[start_idx:end_idx]
    train_clusters_batch = train_clusters[start_idx:end_idx]
    silhouette_avg_train_31 = silhouette_score(train_batch,
  ↳train_clusters_batch)
    silhouette_scores_train.append(silhouette_avg_train_31)

# Compute average silhouette score for training dataset
silhouette_avg_train_31 = sum(silhouette_scores_train) /
  ↳len(silhouette_scores_train)

# Compute silhouette score for testing dataset
test_num_batches = len(test_df3) // batch_size + 1
silhouette_scores_test = []

for i in range(test_num_batches):
    start_idx = i * batch_size
    end_idx = min((i + 1) * batch_size, len(test_df3))
    test_batch = test_df3[start_idx:end_idx]
    test_clusters_batch = test_clusters[start_idx:end_idx]
    silhouette_avg_test_31 = silhouette_score(test_batch, test_clusters_batch)
    silhouette_scores_test.append(silhouette_avg_test_31)

```

```

# Compute average silhouette score for testing dataset
silhouette_avg_test_31 = sum(silhouette_scores_test) /
    len(silhouette_scores_test)

print(f"Silhouette Score (Training dataset): {silhouette_avg_train_31}")
print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_31}")

```

Silhouette Score (Training dataset): 0.6601892574986835

Silhouette Score (Testing dataset): 0.6592176631400798

```

[32]: plt.figure(figsize=(4, 4))

for cluster in range(n_clusters):
    cluster_data = train_df3[train_clusters == cluster]

    plt.scatter(cluster_data['tBodyAcc-mean()-X'],
        cluster_data['tBodyAcc-mean()-Y'], label=f'Cluster {cluster}')

plt.scatter(cluster_centroids[:, 0], cluster_centroids[:, 1], marker='x',
    color='black', s=100, label='Centroids')

plt.title('Clustering Results')
plt.xlabel('GravityAcc-mean()-X')
plt.ylabel('tGravityAcc-max()-X')
plt.legend()
plt.grid(True)
plt.show()

```



$-1 \leq \text{Silhouette score} \leq 1$

A Silhouette score of 1 indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

A Silhouette score of 0 indicates that the object is on or very close to the decision boundary between two neighboring clusters.

A Silhouette score of -1 indicates that the object is poorly matched to its own cluster and well matched to neighboring clusters.

1) Select Best K (Filter method)

```
print(f"Silhouette Score (Training dataset): {silhouette_avg_train_11}") print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_11}")
```

2) Ridge Regression

```
print(f"Silhouette Score (Training dataset): {silhouette_avg_train_21}") print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_21}")
```

3) PCA

```
print(f"Silhouette Score (Training dataset): {silhouette_avg_train_31}") print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_31}")
```

```
[48]: print("\n\n1) Select Best K ( Filter method )")
```

```

print(f"Silhouette Score (Training dataset): {silhouette_avg_train_11}")
print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_11}")

print("\n\n2) Ridge Regression")

print(f"Silhouette Score (Training dataset): {silhouette_avg_train_21}")
print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_21}")

print("\n\n3) PCA")

print(f"Silhouette Score (Training dataset): {silhouette_avg_train_31}")
print(f"Silhouette Score (Testing dataset): {silhouette_avg_test_31}")

```

1) Select Best K (Filter method)
 Silhouette Score (Training dataset): 0.3397089651970844
 Silhouette Score (Testing dataset): 0.3529194692224184

2) Ridge Regression
 Silhouette Score (Training dataset): 0.6952144230232754
 Silhouette Score (Testing dataset): 0.7131016420643701

3) PCA
 Silhouette Score (Training dataset): 0.6601892574986835
 Silhouette Score (Testing dataset): 0.6592176631400798

2 Inference

The Model which was build using the Select best K feature Engineering techniques gives the best Silhouette Score. So ridge regression seems to be the preferred model.

3 Learning Outcomes

1. Understood the working of K-means clustering algorithm.
2. Data preprocessing.
3. Interpretation of clustering results.
4. Understood the activity recognition using sensor data