

**VILNIUS UNIVERSITY**

**ŠIAULIAI ACADEMY**

BACHELOR PROGRAMME SOFTWARE ENGINEERING

**Artificial Intelligence**

**Report on “Clustering” task**

Student: Anna Kutova

Lecturer: Dr. Gintautas Daunys

Šiauliai, 2023

# Used code

*My complete code for this assignment:*

from sklearn.cluster import KMeans, AgglomerativeClustering  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.colors as mcolors  
from sklearn import model\_selection  
from sklearn.decomposition import PCA  
from sklearn.preprocessing import StandardScaler  
  
data = pd.read\_csv('accent-mfcc-data-1.csv')  
X = data.drop('language', axis=1)  
y = data['language']  
  
# Data split  
X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(X, y, test\_size=0.2, random\_state=1)  
  
# Scaling  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
  
# KMeans Clustering  
kmeans = KMeans(n\_clusters=20)  
kmeans.fit(X\_train\_scaled)  
labels\_train = kmeans.labels\_  
labels\_test = kmeans.predict(X\_test\_scaled)  
  
# DataFrame  
df = pd.DataFrame({'Class': y\_test, 'Cluster': labels\_test})  
cluster\_counts = df.groupby(['Cluster', 'Class']).size().unstack(fill\_value=0)  
  
print(cluster\_counts)  
  
  
# Agglomerative Clustering  
agg\_clustering = AgglomerativeClustering(n\_clusters=10, linkage='ward')  
agg\_labels\_train = agg\_clustering.fit\_predict(X\_train\_scaled)  
agg\_labels\_test = agg\_clustering.fit\_predict(X\_test\_scaled)  
  
  
# DataFrame  
df\_agg = pd.DataFrame({'Class': y\_test, 'Cluster': agg\_labels\_test})  
cluster\_counts\_agg = df\_agg.groupby(['Cluster', 'Class']).size().unstack(fill\_value=0)  
  
print("Agglomerative Clustering Results:")  
print(cluster\_counts\_agg)  
  
  
# Visualize results using PCA  
pca = PCA(n\_components=2)  
X\_train\_pca = pca.fit\_transform(X\_train\_scaled)  
X\_test\_pca = pca.transform(X\_test\_scaled)  
  
# Plot KMeans Clustering  
plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.scatter(X\_train\_pca[:, 0], X\_train\_pca[:, 1], c=labels\_train, cmap='magma')  
plt.title('KMeans Clustering - Training Data')  
  
plt.subplot(1, 2, 2)  
plt.scatter(X\_test\_pca[:, 0], X\_test\_pca[:, 1], c=labels\_test, cmap='magma')  
plt.title('KMeans Clustering - Test Data')  
  
plt.show()  
  
# Plot Agglomerative Clustering  
plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.scatter(X\_train\_pca[:, 0], X\_train\_pca[:, 1], c=agg\_labels\_train, cmap='viridis')  
plt.title('Agglomerative Clustering - Training Data')  
  
plt.subplot(1, 2, 2)  
plt.scatter(X\_test\_pca[:, 0], X\_test\_pca[:, 1], c=agg\_labels\_test, cmap='viridis')  
plt.title('Agglomerative Clustering - Test Data')  
  
plt.show()

# Explaining code step by step: selecting labeled dataset; splitting to train and test subsets.

* Importing necessary libraries for clustering and loadeding a dataset of digits:

from sklearn.cluster import KMeans, AgglomerativeClustering  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.colors as mcolors  
from sklearn import model\_selection  
from sklearn.decomposition import PCA  
from sklearn.preprocessing import StandardScaler  
  
data = pd.read\_csv('accent-mfcc-data-1.csv')  
X = data.drop('language', axis=1)  
y = data['language']  
X = digits['data']

* Separation of data into training and test sets:

X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(X, y, test\_size=0.2, random\_state=1)

* Dimensionality reduction using PCA:

pca = PCA(n\_components=2)  
X\_train\_pca = pca.fit\_transform(X\_train\_scaled)  
X\_test\_pca = pca.transform(X\_test\_scaled)

* PCA Model Training:

# Visualize results using PCA  
pca = PCA(n\_components=2)  
X\_train\_pca = pca.fit\_transform(X\_train\_scaled)  
X\_test\_pca = pca.transform(X\_test\_scaled)

# Data split, scaling and training Kmean model. Training PCA model on train subset, reducing number of coordinates to 2. Kmean cluster table and visualization

* Plotting KMeans Clustering:

# Plot KMeans Clustering  
plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.scatter(X\_train\_pca[:, 0], X\_train\_pca[:, 1], c=labels\_train, cmap='magma')  
plt.title('KMeans Clustering - Training Data')  
  
plt.subplot(1, 2, 2)  
plt.scatter(X\_test\_pca[:, 0], X\_test\_pca[:, 1], c=labels\_test, cmap='magma')  
plt.title('KMeans Clustering - Test Data')  
  
plt.show()

*In these lines, we visualize the results of KMeans clustering in a 2D space. The left subplot represents the training data, and the right subplot represents the test data. Each point corresponds to a sample, and colors indicate the cluster assignment.*

* KMeans Clustering:

plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.scatter(X\_train\_pca[:, 0], X\_train\_pca[:, 1], c=labels\_train, cmap='magma')  
plt.title('KMeans Clustering - Training Data')

*Here, KMeans clustering is applied with a fixed number of clusters (in this case, 10) on the scaled training data. The resulting cluster labels for both the training and test sets are obtained.*

* Creating a DataFrame and Counting Samples in Each Cluster:
* df\_agg = pd.DataFrame({'Class': y\_test, 'Cluster': agg\_labels\_test})  
  cluster\_counts\_agg = df\_agg.groupby(['Cluster', 'Class']).size().unstack(fill\_value=0)  
    
  print("Kmeans Clustering Results:")  
  print(cluster\_counts\_agg)

*In this section, a DataFrame (****df****) is created to store the class labels and corresponding cluster labels for the test set. Then, a table (****cluster\_counts****) is generated to count how many samples from each class fall into each cluster. This table is printed to provide insights into the distribution of samples across clusters.*

* KMeans Clustering Table:

*A screenshot of a black screen

Description automatically generated*

* KMeans Clustering Visualisation:

A comparison of a number of dots

Description automatically generated with medium confidence

# Repeat step 3) and 4) with bigger number of clusters than unique class labels.

* Code modifications:

# KMeans Clustering  
kmeans = KMeans(n\_clusters=20)  
kmeans.fit(X\_train\_scaled)  
labels\_train = kmeans.labels\_  
labels\_test = kmeans.predict(X\_test\_scaled)

*Adjustment in the highlighted line instructs the algorithm to utilize 20 clusters during the clustering process, expanding the number of clusters compared to the previous usage where 10 clusters were set*

* Results in Table and in Visualisation:

A black and white screen with numbers

Description automatically generatedA comparison of a number of dots

Description automatically generated with medium confidence

# Agglomerative clustering. TABLE AND VISUALIZATION

* Plotting KMeans Clustering Results:

# Plot Agglomerative Clustering  
plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.scatter(X\_train\_pca[:, 0], X\_train\_pca[:, 1], c=agg\_labels\_train, cmap='viridis')  
plt.title('Agglomerative Clustering - Training Data')  
  
plt.subplot(1, 2, 2)  
plt.scatter(X\_test\_pca[:, 0], X\_test\_pca[:, 1], c=agg\_labels\_test, cmap='viridis')  
plt.title('Agglomerative Clustering - Test Data')  
  
plt.show()

*These lines visualize the results of Agglomerative clustering in a 2D space. The left subplot represents the training data, and the right subplot represents the test data. Each point corresponds to a sample, and colors*

* Agglomerative Clustering:

plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.scatter(X\_train\_pca[:, 0], X\_train\_pca[:, 1], c=agg\_labels\_train, cmap='viridis')  
plt.title('Agglomerative Clustering - Training Data')

*Here, Agglomerative clustering is applied with a specified number of cluster and the linkage method 'ward' on the scaled training data. The resulting cluster labels for both the training and test sets are obtained.*

* Creating a DataFrame and Counting Samples in Each Cluster:
* df\_agg = pd.DataFrame({'Class': y\_test, 'Cluster': agg\_labels\_test})  
  cluster\_counts\_agg = df\_agg.groupby(['Cluster', 'Class']).size().unstack(fill\_value=0)  
    
  print("Agglomerative Clustering Results:")  
  print(cluster\_counts\_agg)

*In this section, a DataFrame (****df\_agg****) is created to store the class labels and corresponding cluster labels for the test set. Then, a table (****cluster\_counts\_agg****) is generated to count how many samples from each class fall into each cluster. This table is printed to provide insights into the distribution of samples across clusters.*

* Agglomerative Clustering Table:

*A screenshot of a black screen

Description automatically generated*

* Agglomerative Clustering Visualisation:

A comparison of a graph

Description automatically generated with medium confidence

# Setting n\_clusters=None and trying different distance threshold values. Searching for threshold and linkage.

n\_clusters=None

* 1st case:

linkage='ward'

distance\_threshold=2)

agg\_clustering = AgglomerativeClustering(n\_clusters=None, linkage='ward', distance\_threshold=2)

A diagram of a cluster of dots

Description automatically generatedA screenshot of a black screen

Description automatically generatedA screenshot of a black screen

Description automatically generatedTable and visualisation:

* 2nd case:

linkage='complete'

distance\_threshold=13)

A comparison of a graph

Description automatically generated with medium confidenceA screenshot of a black screen

Description automatically generatedTable and visualisation:

* 3rd case:

linkage='average'

distance\_threshold=7)

A black screen with white text

Description automatically generatedA chart with yellow dots

Description automatically generatedTable and visualisation:

# Conclusion

In this clustеring assignmеnt, thе goal was to mastеr paramеtеr sеlеction in clustеring algorithms, visualizе clustеrs through dimеnsionality rеduction, and prеsеnt rеsults еffеctivеly. Kеy stеps includеd datasеt sеlеction and splitting, PCA modеl training, K-Mеans clustеring, visualization of clustеrs, еxploration of largеr clustеr numbеrs, Agglomеrativе clustеring with paramеtеr еxploration, and rеsult prеsеntation in a rеport.

Thе analysis еncompassеd using Python codе, prеsеnting clustеr composition tablеs, showcasing clustеr plots aftеr PCA rеduction, and еxplaining thе rеsults. Through this assignmеnt, I'vе got a dееpеr undеrstanding of clustеring tеchniquеs and thеir application in rеal-world scеnarios was gainеd, contributing to a morе comprеhеnsivе skill sеt in data analysis.