|  |
| --- |
|  |
| page1image29306176  Τμήμα Μηχανικών Η/Υ και Πληροφορικής  Πανεπιστήμιο Πατρών  Πολυτεχνική Σχολή  Τομέας Λογικού των Υπολογιστών  Διδάσκοντες: Βασίλειος Μεγαλοοικονόμου, Χρήστος Μακρής  Ακαδημαϊκό Έτος: 2023 – 2024  Ημ / νία Παράδοσης: 09/ 06 / 2024 |

Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

Επιλεγόμενο Μάθημα – CEID\_ΝΕ562

Εργαστηριακή Άσκηση

Εαρινό Εξάμηνο 2024

Χρυσαυγή Πατέλη | 1084513 | up1084513@ac.upatras.gr

Μηλτιάδης Μαντές | 1084661 | up1084661@ac.upatras.gr

ΠΕΡΙΕΧΟΜΕΝΑ

**1 ΕΙΣΑΓΩΓΗ 1.1** Σύνολο Δεδομένων ……………..…………..……..……………………………………………….. 3 **1.2** Περιβάλλον Υλοποίησης & Βιβλιοθήκες …………………………………………………………… 4

**2 ΕΡΩΤΗΜΑ 1Ο 2.1** Περιγραφή …..…………………………………………………………………………………….. 6 **2.2** Υλοποίηση …..…………………………………………………………………………………….. 7  **2.3** Αποτελέσματα ……………………………………………………………………………….…... 13

**3 ΕΡΩΤΗΜΑ 2Ο 3.1** Περιγραφή …………………………………………………………………………………..…… 23 **3.2** Υλοποίηση ……………………………………………………………………………………..…  **3.3** Αποτελέσματα …………………………………………………………………………………..

**4 ΕΡΩΤΗΜΑ 3Ο 4.1** Περιγραφή ……………………………………………………………………………………… **4.2** Υλοποίηση ………………………………………………………………………………………  **4.3** Αποτελέσματα …………………………………………………………………………………..

**5 ΠΑΡΑΡΤΗΜΑ**

1. **ΕΙΣΑΓΩΓΗ**

**1.1 Σύνολο Δεδομένων**

**1.2 Περιβάλλον Υλοποίησης & Βιβλιοθήκες**

**Περιβάλλον Υλοποίησης:**

Ως περιβάλλον υλοποίησης έχουμε επιλέξει το **Google Colab**, το οποίο μας προσφέρει μεγάλη διευκόλυνση όσον αφορά την υλοποίηση και τη ταχύτητα εκτέλεσης κώδικα πάνω σε μεγάλο όγκο δεδομένων. Πιο συγκεκριμένα, προσφέρει δωρεάν πρόσβαση σε *Τ4* *GPU* (Graphics Processing Unit) και *TPU* (Tensor Processing Unit), επιτρέποντας έτσι την ταχύτερη εκτέλεση αλγορίθμων μηχανικής μάθησης και deep learning, το οποίο είναι και το κύριο αντικείμενο αυτής της εργασίας. Άλλωστε σε κάθε κελί ενός Notebook σε **Google Colab** μπορούμε να γράψουμε και να εκτελέσουμε κώδικα Python ακριβώς όπως και σε ένα τοπικό **Jupyter Notebook** χωρίς καμία απολύτως διαφορά. Τέλος, ένας άλλος λόγος που επιλέξαμε το **Google Colab** είναι ότι διευκόλυνε τη συνεργατική υλοποίηση της εργασίας, καθώς μπορούσαμε να μοιραστούμε τα Notebooks μας και να τα επεξεργαστούμε σε πραγματικό χρόνο.

Παρακάτω ακολουθεί και μια σύντομη περιγραφή όλων των βιβλιοθηκών λογισμικού που εγκαταστήσαμε στο περιβάλλον μας για την υλοποίηση της εργασίας:

**Γενικές Βιβλιοθήκες:**

1. **os:** παρέχει διασύνδεση με το λειτουργικό σύστημα, επιτρέποντας την εκτέλεση εντολών για τη διαχείριση αρχείων και καταλόγων. Έτσι, μπορούμε να αλλάξουμε τον τρέχοντα κατάλογο και να διαβάσουμε το περιεχόμενο ενός καταλόγου στο κώδικά μας.
2. **pandas:** είναι μια ισχυρή βιβλιοθήκη για την ανάλυση και τη διαχείριση δεδομένων. Παρέχει δομές δεδομένων όπως DataFrame και Series που διευκολύνουν την επεξεργασία, τον καθαρισμό και την ανάλυση δεδομένων που διαβάζουμε από τα αρχεία CSV του dataset.
3. **numpy:** είναι η βασική βιβλιοθήκη για αριθμητικούς υπολογισμούς στη Python. Παρέχει υποστήριξη για πολυδιάστατους πίνακες και μια πληθώρα μαθηματικών συναρτήσεων για τη λειτουργία πάνω σε αυτούς τους πίνακες.
4. **matplotlib.pyplot:** είναι ένα υποσύνολο της βιβλιοθήκης **matplotlib** και παρέχει ένα σύνολο εντολών που επιτρέπουν τη δημιουργία διαγραμμάτων και γραφημάτων.

**Βιβλιοθήκες 1ου Ερωτήματος:**

1. **seaborn:** είναι μια βιβλιοθήκη για τη στατιστική απεικόνιση των δεδομένων μας που βασίζεται στη **matplotlib**. Παρέχει ευκολότερους τρόπους για τη δημιουργία σύνθετων γραφημάτων. Στο συγκεκριμένο ερώτημα εμείς τη χρησιμοποιούμε για την υλοποίηση γραφημάτων διασποράς και θερμοχάρτες (heatmaps) που προκύπτουν από τη στατιστική επεξεργασία των δεδομένων.

**Βιβλιοθήκες 2ου Ερωτήματος:**

1. **sklearn.metrics:** είναι υποσύνολο της βιβλιοθήκης **sklearn** η οποία χρησιμοποιείται για μηχανική μάθηση καιπαρέχει εργαλεία για την αξιολόγηση της απόδοσης μοντέλων μηχανικής μάθησης. Περιλαμβάνει μετρικές για ταξινόμηση, παλινδρόμηση και ομαδοποίηση, όπως **accuracy\_score**, **precision\_score**, **recall\_score**, **f1\_score**, **confusion matrix** κ.ά.
2. **sklearn.ensemble:** είναι υποσύνολο της βιβλιοθήκης **sklearn** και περιλαμβάνει αλγορίθμους συνόλων (ensemble algorithms) που συνδυάζουν τις προβλέψεις πολλαπλών μοντέλων για τη βελτίωση της απόδοσης και της ακρίβειας. Το μοντέλο που θα αξιοποιήσουμε εμείς από αυτή τη βιβλιοθήκη είναι αυτό του **Random Forest**.
3. **tensorflow:** είναι μια βιβλιοθήκη ανοιχτού κώδικα για τη μηχανική μάθηση και την ανάπτυξη νευρωνικών δικτύων μέσω του *Keras API*. Από το *Keras API* εμείς αξιοποιούμε για την υλοποίηση του **Artificial Neural Network** το Sequential μοντέλο και το στρώμα Dense.
4. **pgmpy.models:** είναι μέρος της βιβλιοθήκης **pgmpy** που χρησιμοποιείται για τη δημιουργία και τη διαχείριση μοντέλων γραφικών πιθανοτήτων (probabilistic graphical models). Αυτά τα μοντέλα περιλαμβάνουν τα **Bayesian Networks** που θα αξιοποιήσουμε εμείς.
5. **pgmpy.estimators:** είναι μέρος της βιβλιοθήκης **pgmpy** και παρέχει μεθόδους για την εκτίμηση παραμέτρων και δομών για τα μοντέλα γραφικών πιθανοτήτων. Χρησιμοποιείται για την εκμάθηση της δομής του δικτύου και των πιθανοτήτων από τα δεδομένα σε ένα **Bayesian Network**. Από αυτές τις μεθόδους εμείς αξιοποιούμε τη **MaximumLikelihoodEstimator**, τη **HillClimbSearch** και τη **BicScore**.
6. **pgmpy.inference:** είναι μέρος της βιβλιοθήκης **pgmpy** και παρέχει εργαλεία και αλγορίθμους για τον υπολογισμό πιθανοτήτων και εκτίμηση μεταβλητών σε πιθανοτικά γραφικά μοντέλα. Η ανάλυση που παρέχει βοηθά στην κατανόηση των συμπερασμάτων που μπορούν να εξαχθούν από τα πιθανοτικά μοντέλα και τις εκτιμήσεις που μπορούν να πραγματοποιηθούν. Για το Bayesian Network αξιοποιήσαμε τη κλάση **VariableElimination** για τον υπολογισμό πιθανοτήτων ή αναμενόμενων τιμών.
7. **sklearn.preprocessing:** είναι υποσύνολο της βιβλιοθήκης **sklearn** και παρέχει εργαλεία για την προεπεξεργασία δεδομένων. Κάποια από αυτά τα εργαλεία τα οποία θα αξιοποιήσουμε είναι η κανονικοποίηση (normalization), και η μετατροπή κατηγοριών σε δυαδικούς δείκτες (one-hot encoding).

**Βιβλιοθήκες 3ου Ερωτήματος:**

1. **sklearn.preprocessing:** είναι υποσύνολο της βιβλιοθήκης **sklearn** και παρέχει εργαλεία για την προεπεξεργασία δεδομένων. Κάποια από αυτά τα εργαλεία τα οποία θα αξιοποιήσουμε είναι η κανονικοποίηση (normalization), και η μετατροπή κατηγοριών σε δυαδικούς δείκτες (one-hot encoding).
2. **sklearn.cluster:** είναι υποσύνολο της βιβλιοθήκης **sklearn** και περιλαμβάνει αλγορίθμους για την ομαδοποίηση (clustering) δεδομένων, όπως **K-means**.
3. **sklearn.mixture:** είναι υποσύνολο της βιβλιοθήκης **sklearn** και παρέχει μοντέλα μείγματος (mixture models), όπως **Gaussian Mixture Model (GMM)** το οποίο και χρησιμοποιήσαμε.
4. **mpl\_toolkits.mplot3d:** είναι μια υποβιβλιοθήκη της **matplotlib** που επιτρέπει τη δημιουργία τρισδιάστατων γραφημάτων. Αξιοποιήθηκε για τη τρισδιάστατη αναπαράσταση των συστάδων στις οποίες χωρίσαμε τα δεδομένα για κάθε έναν από τους δύο αισθητήρες.
5. **minisom:** είναι μια απλή και γρήγορη βιβλιοθήκη για την εκπαίδευση και την εφαρμογή αυτοοργανωμένων χαρτών (Self-Organizing Maps, SOMs). Τα SOMs είναι ένας τύπος τεχνητών νευρωνικών δικτύων που χρησιμοποιούνται για την ομαδοποίηση και τη μείωση της διάστασης των δεδομένων. Μέσω αυτής της βιβλιοθήκης υλοποιήσαμε έτσι τη συσταδοποίηση μέσω **Kohonen Networks**.
6. **ΕΡΩΤΗΜΑ 1Ο**

**2.1 Περιγραφή**

Για να κατανοήσουμε περισσότερο τη δομή του συνόλου δεδομένων που θα μελετήσουμε καθώς και τη κατανομή και τις συσχετίσεις των μετρήσεων των αισθητήρων σε αυτό, πραγματοποιούμε στατιστική επεξεργασία τόσο με **αριθμητικό περιγραφικό** τρόπο όσο και με **γραφικό**.

Όσον αφορά τα **αριθμητικά περιγραφικά μέτρα**, πρόκειται για ποσοτικά μεγέθη που βοηθούν στην περιγραφή της κατανομής του συνόλου δεδομένων με όρους ποσοτικούς. Στη προκειμένη περίπτωση θα χρησιμοποιήσουμε στατιστικά **μέτρα θέσης** (μέση τιμή, διάμεσος, 25ο και 75ο εκατοστημόριο) και **μεταβλητότητας** (εύρος συνόλου δεδομένων ή αλλιώς μέγιστη και ελάχιστη τιμή, τυπική απόκλιση).

**Στατιστικά Μέτρα Θέσης:**

1. **Μέση Τιμή (Mean):** Υπολογίζει τον αριθμητικό μέσο όρο των τιμών κάθε γνωρίσματος.
2. **Διάμεσος (50%):** Η τιμή που χωρίζει τα δεδομένα σε δύο ίσα μέρη όταν αυτά ταξινομούνται.
3. **25ο και 75ο Εκατοστημόριο (Quartiles):** Τα σημεία που χωρίζουν το κατώτερο 25% και 75% των δεδομένων.

**Στατιστικά Μέτρα Μεταβλητότητας:**

1. **Εύρος (Range):** Η διαφορά μεταξύ της μέγιστης και ελάχιστης τιμής.
2. **Τυπική Απόκλιση (Standard Deviation):** Ένα μέτρο της διασποράς των τιμών γύρω από τη μέση τιμή.

Όσον αφορά την **γραφική αναπαράσταση** των δεδομένων αξιοποιούμε ένα **ιστόγραμμα κατανομής συχνοτήτων** κάθε γνωρίσματος, καθώς και ένα **θερμοχάρτη (heatmap)** για τον εντοπισμό της ετεροσυσχέτισης ανάμεσα στα 6 γνωρίσματα του συνόλου δεδομένων. Το ιστόγραμμα συχνοτήτων επιπλέον προσαρμόζεται κατάλληλα για να απεικονίζεται η πυκνότητα πιθανότητας κάθε γνωρίσματος και όχι οι απόλυτες συχνότητες εμφάνισής του. Αυτό σημαίνει ότι το άθροισμα των περιοχών των ράβδων θα είναι ίσο με 1, καθιστώντας το ιστόγραμμα κατάλληλο για τη σύγκριση με κανονικοποιημένες κατανομές. Έχουμε επιλέξει πάνω σε κάθε ιστόγραμμα να εμφανίζεται γραφικά και η αντίστοιχη **κατανομή Gauss** την οποία προσεγγίζει η κάθε συνάρτηση πυκνότητας πιθανότητας, προκειμένου η σύγκριση να είναι πιο κατανοητή. Ακόμη, για τον θερμοχάρτη επιλέγουμε να χρησιμοποιήσουμε τον **συντελεστή συσχέτισης Pearson** για να μετρήσουμε τη γραμμική συσχέτιση μεταξύ των γνωρισμάτων.

Τέλος, εφόσον τα δεδομένα μας μεταβάλλονται και με βάση το χρόνο, πέρα από στατιστική επεξεργασία κάνουμε και απεικόνιση της **χρονοσειράς** κάθε γνωρίσματος. Η χρησιμότητα της χρονοσειράς έγκειται στο ότι μπορούμε να αναγνωρίσουμε πιο εύκολα τάσεις, διακυμάνσεις, μοτίβα και ανωμαλίες στα δεδομένα με τη πάροδο του χρόνου.

Πριν προχωρήσουμε στην υλοποίηση, σημειώνουμε ότι έχουμε πραγματοποιήσει στατιστική επεξεργασία των γνωρισμάτων του συνόλου δεδομένων τόσο συγκεντρωτικά (λαμβάνοντας υπόψη τις μετρήσεις των αισθητήρων από όλους τους συμμετέχοντες ταυτόχρονα), όσο και ατομικά (λαμβάνοντας υπόψη τις μετρήσεις από κάθε συμμετέχοντα ξεχωριστά). Επιπλέον, έχουμε επεξεργαστεί στατιστικά όχι μόνο τα δεδομένα ως προς τους συμμετέχοντες αλλά και ως προς τις διακριτές φυσικές δραστηριότητες που πραγματοποιούν. Άρα συνολικά έχουμε πραγματοποιήσει 3 φάσεις επεξεργασίας: **Συνολική Στατιστική Επεξεργασία**, **Στατιστική Επεξεργασία ανά Συμμετέχοντα** και **Στατιστική Επεξεργασία ανά Δραστηριότητα (label)**.

**2.2 Υλοποίηση 2.2.1 Συνολική Στατιστική Επεξεργασία**

Αρχικά, ορίζουμε το μονοπάτι του φακέλου που περιέχει τα αρχεία CSV και μέσω της **os.chdir( )** αλλάζουμε τον τρέχοντα κατάλογο εργασίας με το μονοπάτι που είναι αποθηκευμένο το dataset, ώστε να μπορεί να διαβάσει το πρόγραμμα τα αρχεία CSV από αυτόν τον φάκελο. Έπειτα, για κάθε αρχείο (δηλαδή για κάθε συμμετέχοντα) αποθηκεύουμε τις στήλες του σε ένα ξεχωριστό DataFrame και έπειτα όλα τα DataFrames από όλα τα αρχεία αποθηκεύονται στη λίστα **dfs**. Μέσω της **pd.concat( )** όλα τα DataFrame ενώνονται σε ένα **merged\_df**, από το οποίο μέσω της **drop( )** αφαιρούμε όλες τις αχρείαστες στήλες. Τέλος, με τη **describe( )** εμφανίζονται στην οθόνη τα βασικά στατιστικά μεγέθη που αναφέραμε πιο πάνω συγκεντρωτικά για όλο το dataset. Έπειτα, δημιουργούμε ένα ιστόγραμμα με τη χρήση της μεθόδου **ax.hist()** καθένα από τα οποία αναπαριστά τη συχνότητα εμφάνισης των διαφορετικών τιμών σε κάθε μία από τις 6 στήλες με τα δεδομένα των αισθητήρων.

import os

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

*# Μονοπάτι για τον φάκελο που περιέχει τα αρχεία CSV*

path = '/content/drive/MyDrive/harth'

os.chdir(path)

*# Πραγματοποίηση πέρασματος σε κάθε αρχείο CSV*

dfs = []

for filename in os.listdir(path):

if filename.endswith('.csv'):

df = pd.read\_csv(os.path.join(path, filename))

dfs.append(df)

*# Συγχώνευση όλων των DataFrame σε ένα*

merged\_df = pd.concat(dfs, ignore\_index=True)

*# Αφαίρεση των αχρείαστων στηλών 'index', 'label' και 'Unnamed: 0'*

merged\_df.drop(['index', 'label', 'Unnamed: 0'], axis=1, inplace=True)

print(merged\_df.describe().T)

Αρχικά, ορίζουμε το μονοπάτι του φακέλου που περιέχει τα αρχεία CSV και μέσω της **os.chdir( )** αλλάζουμε τον Έπειτα, δημιουργούμε ένα ιστόγραμμα με τη χρήση της μεθόδου **ax.hist( )** καθένα από τα οποία αναπαριστά τη συχνότητα εμφάνισης των διαφορετικών τιμών σε κάθε μία από τις 6 στήλες με τα δεδομένα των αισθητήρων. Για κάθε στήλη, προσεγγίζουμε την κατανομή των δεδομένων με μια κανονική κατανομή. Υπολογίζουμε τη μέση τιμή **mu** και την τυπική απόκλιση **sigma** των δεδομένων του **merged\_df** με τις συναρτήσεις **mean( )** και **std( )** και σχεδιάζουμε την κατανομή πιθανότητας της κανονικής κατανομής **p** πάνω από το αντίστοιχο ιστόγραμμα.

*# Αναπαράσταση γραφικών της συχνότητας των τιμών για τις 6 στήλες*

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

fig.suptitle('Appearance Frequency Distributions of Features', fontsize=10)

for ax, column in zip(axes.flatten(), ['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']):

ax.hist(merged\_df[column], bins=35, color='skyblue', edgecolor='black', density=True)

ax.set\_title(f'{column}', fontsize=10)

ax.set\_ylabel('Density', fontsize=8)

ax.tick\_params(axis='both', which='major', labelsize=6)

ax.grid(True)

*# Προσεγγίζουμε τη κατανομή συχνοτήτων με μια Κανονικη Κατανομή*

mu, sigma = merged\_df[column].mean(), merged\_df[column].std()

xmin, xmax = merged\_df[column].min(), merged\_df[column].max()

x = np.linspace(xmin, xmax, 50000)

p = (1 / (sigma \* np.sqrt(2 \* np.pi))) \* np.exp(-(x - mu) \*\* 2 / (2 \* sigma \*\* 2))

ax.plot(x, p, 'k', linewidth=1)

plt.tight\_layout()

plt.show()

Με την εντολή **merged\_df[['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']].corr( )** υπολογίζουμε τον πίνακα συσχέτισης για τις επιλεγμένες στήλες του DataFrame. Η συνάρτηση **corr( )** υπολογίζει το συντελεστή συσχέτισης Pearson για κάθε ζεύγος στηλών. Επίσης, η **sns.heatmap( )** υλοποιεί το θερμοχάρτη με βάση τις τιμές των συσχετίσεων που υπολογίσθηκαν.

*# Δημιουργία του heatmap*

plt.figure(figsize=(10, 6))

sns.heatmap(merged\_df[['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']].corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Heatmap of Feature Correlations')

plt.show()

Με βάση τον θερμοχάρτη που εμφανίστηκε εντοπίζουμε τα 4 ζεύγη γνωρισμάτων με τη μεγαλύτερη συσχέτιση (**thigh\_x και back\_x, thigh\_x και thigh\_z, back\_y και thigh\_y,** **back\_z και thigh\_z**) και με την συνάρτηση **scatter( )** εκτυπώνουμε για κάθε ζεύγος το scatter plot με όλα τα δεδομένα κάθε γνωρίσματος. Αυτή η απεικόνιση μπορεί να βοηθήσει στην αναγνώριση τυχόν γραμμικών σχέσεων ή άλλων μοτίβων μεταξύ των χαρακτηριστικών.

*# Γραφική αναπαράσταση των 4 ζευγαριών από features με τις μεγαλύτερες ετεροσυσχετίσεις*

fig, axs = plt.subplots(2, 2, figsize=(12, 8))

*# Ζευγάρι 1: thigh\_x και back\_x*

axs[0, 0].scatter(merged\_df['thigh\_x'], merged\_df['back\_x'], alpha=0.5)

axs[0, 0].set\_xlabel('thigh\_x')

axs[0, 0].set\_ylabel('back\_x')

axs[0, 0].set\_title('Correlation between thigh\_x and back\_x')

*# Ζευγάρι 2: thigh\_x και thigh\_z*

axs[0, 1].scatter(merged\_df['thigh\_x'], merged\_df['thigh\_z'], alpha=0.5)

axs[0, 1].set\_xlabel('thigh\_x')

axs[0, 1].set\_ylabel('thigh\_z')

axs[0, 1].set\_title('Correlation between thigh\_x and thigh\_z')

*# Ζευγάρι 3: back\_y και thigh\_y*

axs[1, 0].scatter(merged\_df['back\_y'], merged\_df['thigh\_y'], alpha=0.5)

axs[1, 0].set\_xlabel('back\_y')

axs[1, 0].set\_ylabel('thigh\_y')

axs[1, 0].set\_title('Correlation between back\_y and thigh\_y')

*# Ζευγάρι 4: back\_z και thigh\_z*

axs[1, 1].scatter(merged\_df['back\_z'], merged\_df['thigh\_z'], alpha=0.5)

axs[1, 1].set\_xlabel('back\_z')

axs[1, 1].set\_ylabel('thigh\_z')

axs[1, 1].set\_title('Correlation between back\_z and thigh\_z')

plt.tight\_layout()

plt.show()

**2.2.2 Στατιστική Επεξεργασία ανά Συμμετέχοντα**

Τώρα θέλουμε να δημιουργήσουμε και εμφανίσουμε τις χρονοσειρές για κάθε αρχείο CSV που βρίσκεται στο συγκεκριμένο φάκελο με το dataset. Αρχικά, η λίστα **file\_names** δημιουργείται για να αποθηκεύσει τα ονόματα των αρχείων CSV (δηλαδή των συμμετεχόντων). Αφού αποθηκεύσουμε πάλι κάθε αρχείο σε ένα DataFrame μέσω της **pd.to\_datetime( )** μετατρέπουμε τη στήλη **‘timestamp’** κάθε **df** από αλφαριθμητικό σε μορφή datetime. Tέλος, για κάθε χαρακτηριστικό (**back\_x**, **back\_y**, **back\_z**, **thigh\_x**, **thigh\_y**, **thigh\_z**), σχεδιάζεται η χρονοσειρά του μέσω της **plot( )**.

*# Λίστα για τα ονόματα των συμμετεχόντων*

file\_names = []

*# Σχεδιάζουμε τη χρονοσειρά για κάθε αρχείο CSV*

for filename in os.listdir(path):

file\_path = os.path.join(path, filename)

df = pd.read\_csv(file\_path)

*# Χωρίς κατάληξη*

base\_name = os.path.splitext(filename)[0]

*# Μετατροπή του timestamp σε datetime*

df['timestamp'] = pd.to\_datetime(df['timestamp'])

*# Δημιουργία γραφήματος για τα γνωρίσματα*

plt.figure(figsize=(12, 6))

*# Σχεδιάζουμε τη χρονοσειρά για κάθε γνώρισμα*

for column in ['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']:

plt.plot(df['timestamp'], df[column], label=column)

*# Διαμόρφωση γραφήματος*

plt.title(f"Time Series for {base\_name}")

plt.xlabel("Time")

plt.ylabel("Values")

plt.legend()

plt.grid(True)

plt.tight\_layout()

*# Εμφάνιση γραφήματος*

plt.show()

Όμοια τώρα για κάθε αρχείο αποθηκεύουμε τα δεδομένα στο **df** και καλώντας πάνω σε αυτό τη **describe( )** εμφανίζονται τα ίδια στατιστικά μεγέθη για κάθε αρχείο ατομικά αντί από όλα τα αρχεία συνολικά όπως πριν.

*# Πραγματοποίηση πέρασματος στο φάκελο για την εμφάνιση στατιστικών για κάθε αρχείο ξεχωριστά*

for filename in os.listdir(path):

if filename.endswith('.csv'):

df = pd.read\_csv(os.path.join(path, filename))

*# Αφαίρεση της αχρείαστης στήλης 'label'*

df.drop(['label'], axis=1, inplace=True)

*# Χωρίς κατάληξη*

base\_name = os.path.splitext(filename)[0]

print(f"\n Statistic Values for Participant {base\_name}")

print(df.describe().T)

Αρχικά, ορίζουμε ένα λεξικό **mean\_values\_per\_file** που θα αποθηκεύει τις μέσες τιμές των γνωρισμάτων για κάθε συμμετέχοντα.

*# Λίστα με τα ονόματα των συμμετεχόντων*

file\_names = []

*# Λεξικό που θα διατηρεί τις μέσες τιμές των γνωρισμάτων ανά συμμετέχοντα*

mean\_values\_per\_file = {}

*# Λίστα με τα features*

features = ['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']

*# Δημιουργούμε ένα γράφημα για κάθε γνώρισμα*

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

axes = axes.flatten()

for i, feature in enumerate(features):

*# Πραγματοποιούμε πέρασμα σε όλα τα αρχεία CSV στο φάκελο*

for filename in os.listdir(path):

if filename.endswith('.csv'):

*# Χωρίς κατάληξη*

base\_name = os.path.splitext(filename)[0]

file\_names.append(base\_name)

df = pd.read\_csv(os.path.join(path, filename))

*# Υπολογίζουμε τη μέση τιμή του γνωρίσματος για το συγκεκριμένο συμμετέχοντα*

mean\_value = np.mean(df[feature])

*# Προσθέτουμε τη μέση τιμή στο λεξικό*

if base\_name not in mean\_values\_per\_file:

mean\_values\_per\_file[base\_name] = {}

mean\_values\_per\_file[base\_name][feature] = mean\_value

*# Ταξινομούμε τα ονόματα των συμμετεχόντων για να διατηρήσουμε την ίδια σειρά στο γράφημα*

sorted\_file\_names = sorted(mean\_values\_per\_file.keys())

x\_values = range(len(sorted\_file\_names))

y\_values = [mean\_values\_per\_file[participant][feature] for participant in sorted\_file\_names]

axes[i].plot(x\_values, y\_values, '-o', color='blue')

*# Υπολογίζουμε τη συνολική μέση τιμή του γνωρίσματος από όλα τα αρχεία*

total\_mean\_value = np.mean([mean\_values\_per\_file[participant][feature] for participant in sorted\_file\_names])

*# Δημιουργούμε την οριζόντια γραμμή για την συνολική μέση τιμή*

axes[i].axhline(y=total\_mean\_value, color='orange', linestyle='--', label='Overall Mean')

*# Προσθήκη τίτλου και ετικέτας*

axes[i].set\_title(f'Mean Value of Feature {feature}')

axes[i].set\_xlabel('Participants')

axes[i].set\_ylabel('Mean Value')

*# Χρησιμοποιούμε τις ονομασίες των συμμετεχόντων ως ticks στον άξονα x*

axes[i].set\_xticks(range(len(sorted\_file\_names)))

axes[i].set\_xticklabels(sorted\_file\_names, rotation=45)

*# Προσθέτουμε το κάθετο διαγώνιο και εμφανίζουμε το γράφημα*

axes[i].grid(True)

plt.tight\_layout()

plt.show()

**2.2.3 Στατιστική Επεξεργασία ανά Δραστηριότητα**

*# Λίστα με τα χαρακτηριστικά*

features = ['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']

*# Λίστα με τα labels*

labels = [1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 130, 140]

*# Δημιουργία λεξικού για τα DataFrames*

dfs\_per\_label = {label: pd.DataFrame(columns=features) for label in labels}

*# Πραγματοποιούμε πέρασμα σε όλα τα αρχεία CSV στο φάκελο*

for filename in os.listdir(path):

if filename.endswith('.csv'):

df = pd.read\_csv(os.path.join(path, filename))

for label in labels:

label\_df = df[df['label'] == label]

*# Αν υπάρχουν δεδομένα για το συγκεκριμένο label, τα προσθέτουμε στο αντίστοιχο DataFrame*

if not label\_df.empty:

dfs\_per\_label[label] = pd.concat([dfs\_per\_label[label], label\_df[features]], ignore\_index=True)

*# Τυπώνουμε τα στατιστικά για κάθε label*

for label, df in dfs\_per\_label.items():

print(f"Statistics for Label {label}:")

print(df.describe().T)

print("\n")

*# Για κάθε label δημιουργούμε τα γραφήματα*

fig, axes = plt.subplots(4, 3, figsize=(20, 15))

axes = axes.flatten()

for i, (label, df) in enumerate(dfs\_per\_label.items()):

*# Υπολογίζουμε τις μέσες τιμές για κάθε γνώρισμα*

mean\_values = df.mean()

*# Υπολογίζουμε τη συνολική μέση τιμή*

overall\_mean = mean\_values.mean()

*# Δημιουργία γραφήματος*

sns.lineplot(x=features, y=mean\_values, color='blue', marker='o', ax=axes[i])

axes[i].axhline(y=overall\_mean, color='orange', linestyle='--', label='Overall Mean')

*# Προσθήκη τίτλου και ετικετών*

axes[i].set\_title(f'Mean Values of Features for Label {label}')

axes[i].set\_xlabel('Features')

axes[i].set\_ylabel('Mean Value')

axes[i].grid(True)

plt.tight\_layout()

plt.show()

**2.3 Αποτελέσματα 2.3.1 Συνολική Στατιστική Επεξεργασία**

Παρακάτω φαίνονται τα συνολικά στατιστικά μεγέθη για κάθε ένα από τα 6 γνωρίσματα που μελετάμε:

count mean std min 25% 50% \

back\_x 6461328.0 -0.884957 0.377592 -8.000000 -1.002393 -0.974900

back\_y 6461328.0 -0.013261 0.231171 -4.307617 -0.083129 0.002594

back\_z 6461328.0 -0.169378 0.364738 -6.574463 -0.372070 -0.137451

thigh\_x 6461328.0 -0.594888 0.626347 -8.000000 -0.974211 -0.421731

thigh\_y 6461328.0 0.020877 0.388451 -7.997314 -0.100087 0.032629

thigh\_z 6461328.0 0.374916 0.736098 -8.000000 -0.155714 0.700439

75% max

back\_x -0.812303 2.291708

back\_y 0.072510 6.491943

back\_z 0.046473 4.909483

thigh\_x -0.167876 7.999756

thigh\_y 0.154951 7.999756

thigh\_z 0.948675 8.406235

**Συνολικά Συμπεράσματα:**

* Όλα τα χαρακτηριστικά έχουν τον ίδιο αριθμό δειγμάτων, δηλαδή 6.461.328. Αυτό δείχνει ότι δεν υπάρχουν ελλείποντα δεδομένα για κανένα από τα χαρακτηριστικά. Οι μέσες τιμές δείχνουν ότι τα δεδομένα είναι σχετικά συγκεντρωμένα γύρω από το μηδέν, με κάποιες μικρές αποκλίσεις. Το χαρακτηριστικό **thigh\_z** έχει τη μεγαλύτερη μέση τιμή (0.374916), ενώ το **back\_x** έχει την πιο αρνητική μέση τιμή (-0.884957). Η τυπική απόκλιση δείχνει πόσο διασπασμένα είναι τα δεδομένα από τη μέση τιμή. Το χαρακτηριστικό **thigh\_z** έχει τη μεγαλύτερη διασπορά (0.736098), ενώ το **back\_y** έχει τη μικρότερη (0.231171).
* Ακόμα, τα χαρακτηριστικά **back\_x**, **thigh\_x**, **thigh\_y** και **thigh\_z** έχουν ελάχιστες τιμές κοντά στο -8, που υποδηλώνουν ακραίες αρνητικές τιμές. Το **back\_x** και το **back\_z** έχουν ακραίες θετικές τιμές με μέγιστα 2.291708 και 4.909483 αντίστοιχα, ενώ το **thigh\_x**, **thigh\_y** και **thigh\_z** έχουν μέγιστες τιμές κοντά στο 8.
* Οι τιμές του 25% και του 75% για τα περισσότερα χαρακτηριστικά είναι σχετικά κοντά στο μηδέν, δείχνοντας ότι τα δεδομένα είναι σχετικά συγκεντρωμένα γύρω από τη διάμεσο (50%).

**Σύνθετα Συμπεράσματα:**

* Τα δεδομένα των χαρακτηριστικών **back\_x**, **back\_z** και **thigh\_x** φαίνεται να έχουν πιο αρνητική συμμετρία, καθώς οι μέσες τιμές τους είναι αρνητικές και οι ελάχιστες τιμές τους είναι πολύ χαμηλές. Αντίθετα, τα χαρακτηριστικά **thigh\_y** και **thigh\_z** δείχνουν μια θετική κλίση, με μέσες τιμές θετικές και ελάχιστες τιμές αρνητικές.
* Το χαρακτηριστικό **thigh\_z** έχει τη μεγαλύτερη διασπορά, που δείχνει μεγαλύτερη μεταβλητότητα στις μετρήσεις του. Αντίθετα, το χαρακτηριστικό **back\_y** έχει τη μικρότερη διασπορά, υποδεικνύοντας σταθερότητα στις μετρήσεις του.

Παρακάτω φαίνονται οι γραφικές με τη πυκνότητα πιθανότητας και τη κατανομή που προσεγγίζει κάθε ένα από τα 6 γνωρίσματα που μελετάμε:

Εικόνα που περιέχει γραμμή, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα**Κατανομή Τιμών Αισθητήρα "thigh":**

* Όλες οι κατανομές είναι κεντραρισμένες γύρω από το μηδέν.
* Η κατανομή των τιμών στις διαστάσεις **y** και **z** είναι πιο συγκεντρωμένη γύρω από το μηδέν σε σχέση με την διάσταση **x**.

Εικόνα που περιέχει γραμμή, διάγραμμα, γράφημα, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα**Κατανομή Τιμών Αισθητήρα "back":**

* Οι κατανομές είναι επίσης όλες κεντραρισμένες γύρω από το μηδέν.
* Η κατανομή των τιμών στις διαστάσεις **y** και **z** είναι πιο συγκεντρωμένη γύρω από το μηδέν σε σχέση με την διάσταση **x**.

**Συγκριτική Ανάλυση:**

Και για τα δύο μέρη του σώματος οι κατανομές των τιμών στις διαστάσεις **y** και **z** είναι πιο συγκεντρωμένες και στενότερες γύρω από το μηδέν, υποδεικνύοντας μικρότερη διακύμανση (μεγαλύτερη σταθερότητα) σε αυτές τις διαστάσεις. Αντίθετα, οι κατανομές των τιμών στη διάσταση **x** είναι ελαφρώς πιο πλατιές, υποδεικνύοντας μεγαλύτερη διακύμανση στις μετρήσεις. Συνοψίζοντας, οι κατανομές των τιμών και για **thigh** και για **back** είναι κατά προσέγγιση κανονικές και συμμετρικές, με μικρές αποκλίσεις που μπορεί να υπάρχουν λόγω φυσιολογικής διακύμανσης των δεδομένων. Οι κατανομές δεν φαίνονται να έχουν πολλά ακραία σημεία (outliers), γεγονός που υποδηλώνει ότι τα δεδομένα είναι σχετικά ομοιογενή.

Τέλος, βλέπουμε το θερμοχάρτη και τις γραφικές παραστάσεις συσχέτισης των δεδομένων για κάθε ένα από τα 4 πιο ισχυρά συσχετιζόμενα ζεύγη γνωρισμάτων:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, τετράγωνο, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

**Συμπεράσματα:**

* Τα γνωρίσματα **thigh\_x** και **back\_x** έχουν συντελεστή συσχέτισης 0.44, υποδεικνύοντας μια μέτρια θετική συσχέτιση. Αυτό σημαίνει ότι καθώς αυξάνεται η τιμή του **thigh\_x**, τείνει να αυξάνεται και η τιμή του **back\_x**.
* Τα γνωρίσματα **thigh\_x** και **thigh\_z** έχουν συντελεστή συσχέτισης 0.43, υποδεικνύοντας επίσης μια μέτρια θετική συσχέτιση.
* Τα γνωρίσματα **thigh\_y** και **thigh\_z** έχουν συντελεστή συσχέτισης 0.15, υποδεικνύοντας μια χαμηλή αλλά θετική συσχέτιση.
* Τα γνωρίσματα **back\_y** και **thigh\_y** έχουν συντελεστή συσχέτισης -0.19, υποδεικνύοντας μια χαμηλή αρνητική συσχέτιση.
* Τα γνωρίσματα **back\_z** και **thigh\_z** έχουν συντελεστή συσχέτισης -0.17, υποδεικνύοντας μια χαμηλή αρνητική συσχέτιση.
* Εικόνα που περιέχει στιγμιότυπο οθόνης, διάγραμμα, γράφημα

  Περιγραφή που δημιουργήθηκε αυτόματαΟι περισσότερες από τις υπόλοιπες συσχετίσεις είναι κοντά στο μηδέν, υποδηλώνοντας ότι δεν υπάρχει σημαντική γραμμική συσχέτιση μεταξύ των χαρακτηριστικών αυτών.

Εικόνα που περιέχει στιγμιότυπο οθόνης, διάγραμμα, γράφημα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

**2.3.2 Στατιστική Επεξεργασία ανά Συμμετέχοντα**

Παρακάτω φαίνονται ενδεικτικά τα στατιστικά μεγέθη για 3 συμμετέχοντες που μελετάμε:

Statistic Values for Participant S008

count mean std min 25% 50% 75% \

back\_x 418989.0 -0.920351 0.130877 -3.066853 -0.998823 -0.972054 -0.826290

back\_y 418989.0 0.040018 0.107516 -1.209330 -0.013310 0.048302 0.101082

back\_z 418989.0 -0.326746 0.297591 -0.960136 -0.580225 -0.297335 -0.143575

thigh\_x 418989.0 -0.332798 0.463427 -5.922062 -0.882999 -0.085027 -0.005905

thigh\_y 418989.0 0.050361 0.218070 -2.857320 -0.088989 0.068947 0.166770

thigh\_z 418989.0 0.656598 0.523197 -4.233158 0.256630 0.950087 0.994429

max

back\_x 0.873471

back\_y 1.255642

back\_z 1.872940

thigh\_x 2.312848

thigh\_y 4.809320

thigh\_z 5.324356

Statistic Values for Participant S010

count mean std min 25% 50% 75% \

back\_x 351649.0 -1.019898 0.186093 -2.365137 -1.049960 -1.015238 -0.974607

back\_y 351649.0 -0.002224 0.089664 -0.891766 -0.054221 -0.000030 0.047685

back\_z 351649.0 -0.014543 0.155128 -1.120693 -0.085287 -0.000642 0.071708

thigh\_x 351649.0 -0.905389 0.409359 -4.449651 -1.035450 -0.989746 -0.843884

thigh\_y 351649.0 0.017624 0.262037 -3.333658 -0.097479 0.020164 0.115373

thigh\_z 351649.0 0.146144 0.542611 -4.198951 -0.114183 0.022649 0.313726

max

back\_x -0.309204

back\_y 0.994556

back\_z 1.586732

thigh\_x 1.372425

thigh\_y 2.662940

thigh\_z 5.062087

Statistic Values for Participant S012

count mean std min 25% 50% 75% \

back\_x 382414.0 -0.915515 0.242906 -3.810360 -1.000070 -0.987624 -0.971043

back\_y 382414.0 0.005064 0.137488 -1.506640 -0.035278 0.023321 0.073108

back\_z 382414.0 -0.130062 0.290484 -1.343093 -0.151946 -0.058455 0.027089

thigh\_x 382414.0 -0.321236 0.350287 -6.547333 -0.451028 -0.165490 -0.103086

thigh\_y 382414.0 0.011753 0.204115 -3.652872 -0.121605 0.049881 0.131408

thigh\_z 382414.0 0.809352 0.395776 -2.656861 0.879231 0.978771 0.999394

max

back\_x 0.511867

back\_y 1.758769

back\_z 2.035369

thigh\_x 2.828190

thigh\_y 4.131459

thigh\_z 3.386593

**Συνολικά Συμπεράσματα:**

* Για τα περισσότερα χαρακτηριστικά, οι μέσες τιμές των συμμετεχόντων είναι κοντά στις συνολικές μέσες τιμές, αλλά με μικρότερες τυπικές αποκλίσεις, υποδηλώνοντας ότι τα δεδομένα των συμμετεχόντων είναι λιγότερο διασκορπισμένα σε σχέση με τα συνολικά δεδομένα.
* Υπάρχουν μερικές διαφορές στις μέσες τιμές μεταξύ των συμμετεχόντων και των συνολικών δεδομένων, οι οποίες μπορεί να υποδηλώνουν συγκεκριμένες ιδιαιτερότητες στις κινήσεις ή τις στάσεις των συμμετεχόντων.
* Οι τιμές των χαρακτηριστικών **back\_x**, **back\_y** και **back\_z** για τους συμμετέχοντες φαίνεται να έχουν λιγότερες ακραίες τιμές σε σχέση με τα συνολικά δεδομένα.

Με βάση τα παραπάνω, μπορούμε να συμπεράνουμε ότι οι συμμετέχοντες εμφανίζουν αρκετή ομοιογένεια στις μετρήσεις τους, αλλά υπάρχουν και συγκεκριμένες διαφορές που μπορεί να χρειάζονται περαιτέρω διερεύνηση.

Εικόνα που περιέχει κείμενο, γραμμή, γράφημα, αριθμός

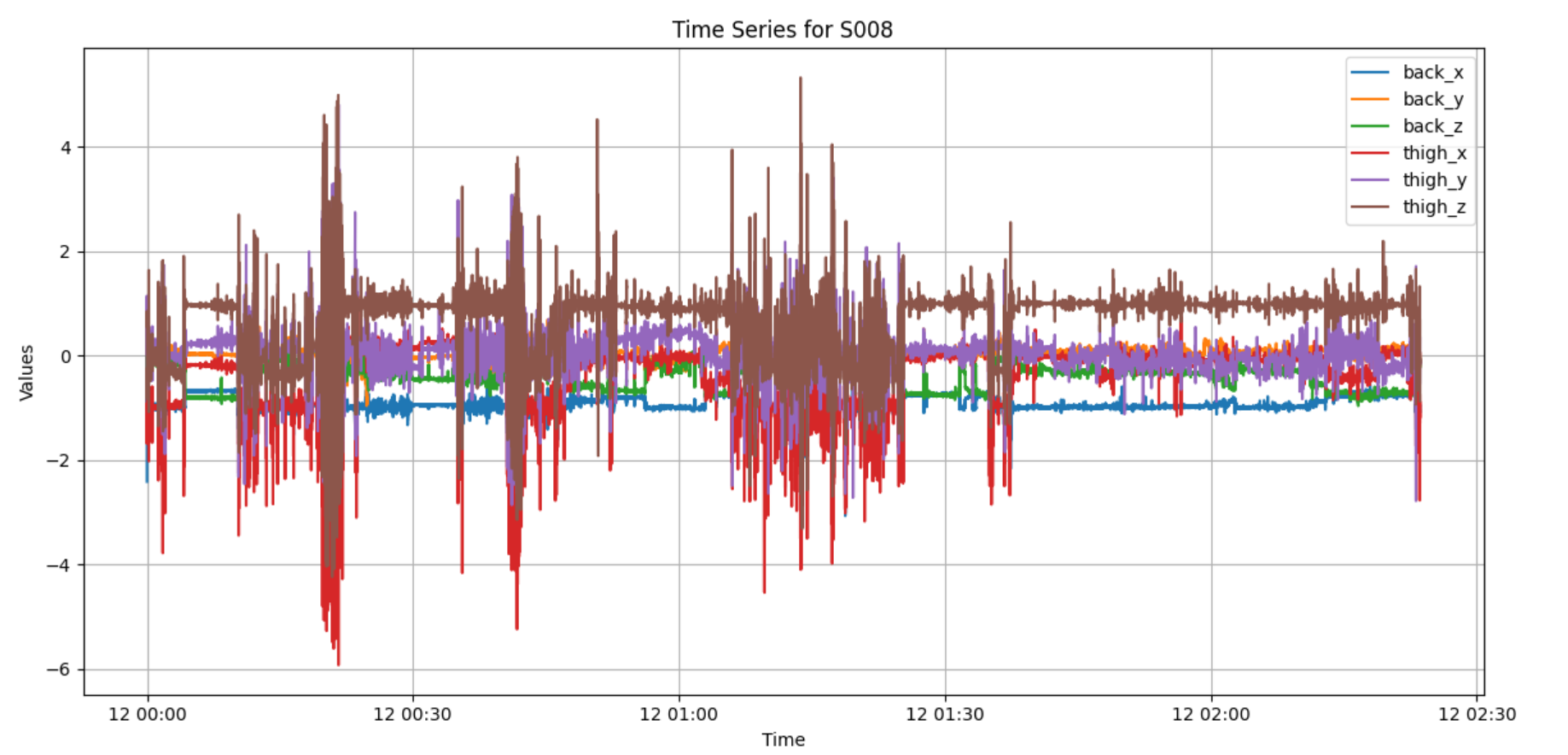
Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει γραμμή, διάγραμμα, κείμενο, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματαΕπίσης, βλέπουμε συνολικά τη μεταβολή της μέσης τιμής κάθε γνωρίσματος αναλογικά με κάθε συμμετέχοντα και την απόκλιση της από τη συνολική μέση τιμή που υπολογίσαμε πριν:

**Συνολικά Συμπεράσματα:**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Τέλος, βλέπουμε ενδεικτικά πάλι τις χρονοσειρές για τους ίδιους 3 συμμετέχοντες και τη μεταβολή κάθε γνωρίσματος τους σε σχέση με το χρόνο μετρήσεων:

**Εικόνα που περιέχει γραμμή, κείμενο, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα Συνολικά Συμπεράσματα:**

**2.3.3 Στατιστική Επεξεργασία ανά Δραστηριότητα**

Παρακάτω φαίνονται ενδεικτικά τα στατιστικά μεγέθη για τις πρώτες 3 δραστηριότητες που μελετάμε:

Statistics for Label 1:

count mean std min 25% 50% \

back\_x 1197155.0 -0.992566 0.311378 -7.974365 -1.158848 -0.975721

back\_y 1197155.0 -0.038755 0.190476 -3.016498 -0.141057 -0.030273

back\_z 1197155.0 -0.137808 0.287737 -3.827393 -0.278809 -0.127185

thigh\_x 1197155.0 -1.056683 0.639900 -8.000000 -1.314697 -0.994395

thigh\_y 1197155.0 -0.023477 0.536738 -7.997314 -0.200704 -0.000487

thigh\_z 1197155.0 -0.074345 0.721997 -8.000000 -0.372559 -0.095456

75% max

back\_x -0.803997 2.291708

back\_y 0.070179 3.256592

back\_z 0.044074 2.896858

thigh\_x -0.808350 4.272705

thigh\_y 0.172181 5.979248

thigh\_z 0.216064 6.897688

Statistics for Label 2:

count mean std min 25% 50% 75% \

back\_x 291356.0 -0.965280 1.113858 -8.000000 -1.871094 -0.858643 0.040771

back\_y 291356.0 -0.076626 0.407701 -4.307617 -0.278076 -0.070312 0.121338

back\_z 291356.0 -0.259829 0.451772 -6.574463 -0.430176 -0.225586 -0.022217

thigh\_x 291356.0 -1.246811 1.438550 -8.000000 -2.219727 -1.141688 -0.265869

thigh\_y 291356.0 -0.164790 0.898353 -7.929199 -0.680930 -0.142822 0.338391

thigh\_z 291356.0 -0.140530 1.381836 -8.000000 -0.871582 -0.169678 0.560547

max

back\_x 1.698069

back\_y 6.491943

back\_z 3.306308

thigh\_x 7.999756

thigh\_y 7.999756

thigh\_z 8.406235

Statistics for Label 3:

count mean std min 25% 50% 75% \

back\_x 254839.0 -0.982356 0.106649 -4.255035 -1.015621 -0.989241 -0.954638

back\_y 254839.0 -0.022316 0.158300 -1.694547 -0.122845 -0.012600 0.079404

back\_z 254839.0 -0.031349 0.208112 -2.093994 -0.159320 -0.027011 0.096443

thigh\_x 254839.0 -0.974374 0.192159 -4.853268 -1.009913 -0.976833 -0.935715

thigh\_y 254839.0 0.020759 0.226300 -5.268984 -0.085332 0.031040 0.136107

thigh\_z 254839.0 -0.068220 0.294548 -5.178711 -0.244184 -0.101230 0.063019

max

back\_x 1.849398

back\_y 1.974854

back\_z 1.754575

thigh\_x 1.664554

thigh\_y 3.755050

thigh\_z 4.701908

**Συνολικά Συμπεράσματα:**

* Η δραστηριότητα 2 (**running**) δείχνει να έχει μεγαλύτερη διακύμανση στα περισσότερα χαρακτηριστικά, όπως υποδεικνύεται από τις υψηλότερες τιμές τυπικής απόκλισης. Επίσης, έχει χαμηλότερες μέσες τιμές σε πολλά χαρακτηριστικά σε σύγκριση με τις δραστηριότητες 1 (**walking**) και 3 (**shuffling**), υποδεικνύοντας πιθανές διαφορές στην κλίση ή τη στάση του σώματος των συμμετεχόντων που ανήκουν σε αυτήν την κατηγορία.
* Οι μέσες τιμές των χαρακτηριστικών για τις δραστηριότητες 1 και 3 (**walking** και **shuffling**) είναι πιο κοντά στη συνολική μέση τιμή, υποδεικνύοντας ότι οι μετρήσεις τους είναι πιο αντιπροσωπευτικές του συνολικού συνόλου.

Αυτές οι διαφορές είναι αναμενόμενες καθώς κάθε φυσική δραστηριότητα απαιτεί διαφορετικές κινήσεις και στάσεις του σώματος.

**Εικόνα που περιέχει κείμενο, γραμμή, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει γραμμή, διάγραμμα, γράφημα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα**Τέλος, βλέπουμε ενδεικτικά πάλι τις χρονοσειρές για τους ίδιους 3 συμμετέχοντες και τη μεταβολή κάθε γνωρίσματος τους σε σχέση με το χρονικό διάστημα μετρήσεων:

1. **Ερώτημα 2ο**

**3.1 Περιγραφή**

**Lag Features**

Στην υλοποίηση μας έχουμε επιλέξει να χρησιμοποιήσουμε τα lag features, είναι χαρακτηριστικά τα οποία χρησιμοποιούνται στην ανάλυση χρονοσειρών, καθώς προσθέτουν πληροφορία σχετικά με τις παρελθοντικές τιμές και έτσι μας επιτρέπουν να δούμε πως οι τιμές αυτές επηρεάζουν τις μελλοντικές. Επιπλέον στα μοντέλα μηχανικής μάθησης που χρησιμοποιούν χρονοσειρές είναι ζωτικής σημασίας, καθώς χρησιμοποιούνται σαν είσοδος, αυξάνοντας τα δεδομένα που χρησιμοποιούνται για πρόβλεψη και έτσι βοηθούν στην κατασκευή ενός μοντέλου που λαμβάνει υπόψη τη συμπεριφορά του παρελθόντος για την πρόβλεψη μελλοντικών γεγονότων.

**Artificial Neural Network**

Για την υλοποίηση του νευρωνικού δικτύου επιλέξαμε να χρησιμοποιήσουμε το **Artificial Neural Network**. Το artificial neural network περιέχει τεχνητούς νευρώνες που ονομάζονται Units. Τα units είναι τοποθετημένα σε μια σειρά από επίπεδα που όλα μαζί αποτελούν το νευρωνικό δίκτυο.

Αποτελείται από 3 επίπεδα:

1. **Είσοδο (input layer):** το στρώμα εισόδου λαμβάνει δεδομένα από τον εξωτερικό κόσμο τα οποία το νευρωνικό δίκτυο πρέπει να αναλύσει ή να μάθει.
2. **Κρυφά επίπεδα (hidden layers):** ένα ή περισσότερα επίπεδα νευρώνων τα οποία επεξεργάζονται την είσοδο και την μτατρέπουν σε δεδομένα που είναι σημαντικά για την έξοδο.
3. **Έξοδος (outout layer):** παράγει την τελική πρόβλεψη.

Τα units συνδέονται μεταξύ τους από το ένα επίπεδο στο άλλο. Κάθε μια από αυτές τις συνδέσεις έχει βάρη που καθορίζουν την επιρροή του ενός Unit σε άλλο unit. Καθώς τα δεδομένα μεταφέρονται από το ένα unit στο άλλο, το νευρωνικό δίκτυο μαθαίνει όλο και περισσότερα για τα δεδομένα, τα οποία τελικά καταλήγουν σε μια έξοδο από το επίπεδο εξόδου.

**Κωδικοποίηση Δεδομένων**

Επιλέγουμε να εφαρμόσουμε κωδικοποίηση **one-hot encoding** στη στήλη labels που είναι και η στήλη εξόδου που θέλουμε να προβλέψει το μοντέλο, πριν αρχίσουμε την εκπαίδευσή του. Η one-hot encoding κωδικοποιεί τα δεδομένα σε δυαδικά διανύσματα, για κάθε δραστηριότητα δημιουργείται ένα διάνυσμα που έχει μήκος ίσο με τον αριθμό των δραστηριοτήτων και το διάνυσμα έχει την τιμή 1 στην θέση που αντιστοιχεί στην δραστηριότητα και 0 σε όλες τις άλλες.

**Δημιουργία Κρυφών Επιπέδων και Εξόδου**

Για την δημιουργία των κρυφών επιπέδων θα χρησιμοποιήσουμε την ενεργοποίηση ReLU, η οποία ελέγχει αν η είσοδος x είναι θετική αν είναι τότε η ReLU επιστρέφει την ίδια είσοδο, ενώ αν είναι αρνητική, η ReLU επιστρέφει το μηδέν. Ενώ για την δημιουργία της εξόδου θα επιλέξουμε την softmax καθώς χρησιμοποιούμε one hot encoding και η softmax δέχεται σαν είσοδο ένα διάνυσμα εισόδου και επιστρέφει ένα διάνυσμα εξόδου που αντιπροσωπεύει μια κατανομή πιθανοτήτων πάνω στις διάφορες κατηγορίες.

**Compiling και Fitting Μοντέλου**

Kατά την εκπαίδευση του μοντέλου μας θα επιλέξουμε να χρησιμοποιήσουμε και την συνάρτηση απώλειας **categorical crossentropy**, η οποία μετρά πόσο καλά αποδίδει το μοντέλο μας στο να προβλέπει τις δραστηριότητες για τα δεδομένα εισόδου. Επιλέγουμε την συγκεκριμένη συνάρτηση, καθώς τα δεδομένα μας έχουν κωδικοποιηθεί με one-hot encoding, οπότε η συνάρτηση αυτή θα υπολογίζει την απόσταση μεταξύ των προβλεπόμενων πιθανοτήτων και των πραγματικών τιμών και αποσκοπεί στην μείωση αυτής της απόστασης. Ουσιαστικά υπολογίζει τη διαφορά μεταξύ του πραγματικού One-hot encoded διανύσματος και του διανύσματος πιθανοτήτων του μοντέλου, αυτό γίνεται για κάθε δείγμα στο σύνολο των δεδομένων. Επίσης, θα χρησιμοποιήσουμε τον αλγόριθμο βελτιστοποίησης **Adam**, ο οποίος προσφέρει ταχύτητα και αποδοτικότητα σε μεγάλα σύνολα δεδομένων. Επιπλέον για την εκπαίδευση του νευρωνικού δικτύου θα χρησιμοποιήσουμε την μέθοδο **batch training**. Η μέθοδος αυτή ορίζει batch\_size αυτό υποδηλώνει ότι τα δεδομένα εκπαίδευσης θα χωριστούν σε μικρότερα υποσύνολα batches και το καθένα batch\_size δείγματα. Επιπλέον, θα πρέπει να ορίσουμε τον αριθμό των **epochs** τα οποία σηματοδοτούν πόσες φορές το μοντέλο θα διατρέξει όλο το σετ δεδομένων. Το μοντέλο προσαρμόζει τα βάρη του μετά από κάθε batch, αντί να περιμένει να δει όλα τα δείγματα σε μια εποχή. Τέλος, το **verbose** καθορίζει τα επίπεδα εξόδου που θα εμφανίζονται κατά τη διάρκεια της εκπαίδευσης και εμείς θα το θέσουμε με 0, οπότε δεν θα εμφανίζεται τίποτα.

**Bayesian Network**

**Βελτιστοποίηση Μοντέλου**

Για την βελτίωση του μοντέλου θα χρησιμοποιηθεί η μέθοδος αναζήτησης **HillClimbSearch**, η οποία χρησιμοποιεί το neighborhood search optimization, ώστε να βρει την βέλτιστη δομή ενός Bayesian Network. Πιο συγκεκριμένα, για να γίνει η αναζήτηση ξεκινάει με μια τυχαία δομή του δικτύου, εξετάζει τις γειτονικές δομές της τρέχουσας και υπολογίζει τη βαθμολογία κάθε γειτονικής δομής χρησιμοποιώντας την μετρική αξιολόγησης **BicScore** και αν προκύψει καλύτερη βαθμολογία από την τρέχουσα, η διαδικασία επαναλαμβάνεται με την καινούρια δομή μέχρι να βρεθεί η βέλτιστη δομή.

**Διακριτοποίηση Χαρακτηριστικών**

Για την διακριτοποίηση θα χρησιμοποιήσουμε το **KBinsDiscretizer,** ο οποίος μετατρέπει τις συνεχείς μεταβλητές σε διακριτές κατηγορίες. Χρησιμοποιείται διότι με διακριτές τιμές είναι πιο εύκολο να υπολογιστούν οι πιθανότητες και να εκτιμηθούν. Επίσης, θα εφαρμόσουμε την στρατηγική **uniform** που θα χωρίσει το εύρος των δεδομένων σε ίσα διαστήματα, δηλαδή κάθε κατηγορία θα καλύπτει το ίδιο εύρος τιμών από το εύρος δεδομένων. Επιπλέον, με το **bins** θα ορίσουμε τον αριθμό των κατηγοριών θα κωδικοποιήσουμε τις διακριτές τιμές ως ακέραιες με το **original.**

**3.2 Υλοποίηση**

**Lag Features**

Αρχικά, δημιουργήθηκε η συνάρτηση **create\_lags(),** η οποία θα χρησιμοποιηθεί και για τις 3 υλοποιήσεις (Neural Network, Bayesian Network, Random Forest). Δημιουργούμε μια λίστα **cols\_to\_shif**t στην οποία αποθηκεύουμε τα ονόματα των στηλών για τις οποίες θέλουμε να δημιουργήσουμε τα lag, δηλαδή όλες εκτός από την στήλη timestamp και label. Στην συνέχεια δημιουργούμε ένα λεξικό **lagged\_data** για κάθε στήλη που χρησιμοποιούμε, το οποίο περιέχει ως κλειδιά τα ονόματα των νέων στηλών που θα δημιουργηθούν, και σαν τιμές εχει τα δεδομένα των στηλών μετατοπισμένα κατά lag χρονικά βήματα. Από το παραπάνω λεξικό δημιουργούμε ένα dataframe **lagged\_df.** Επιπλέον, με την χρήση της **pd.concat** ενώνουμε το αρχικό dataframe με το lagged\_df στο **df.** Η ένωση πραγματοποιείται κατά μήκος των στηλών. Επίσης, η μετατόπιση των δεδομένω στις πρώτες γραμμές του dataframe θα δημιουργεί NaN τιμές, καθώς δεν υπάρχουν τιμές για ναμετατοπιστούν. Οπότε αντικαθιστούμε τις NaN τιμές με 0 διασφαλίζοντας έτσι ότι το dataframe δεν θα περιέχει κενά δεδομένα. Η συνάρτηση επιστρέφει το τελικό dataframe **df.**

import os

import pandas as pd

import numpy as np

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

def create\_lags(df, lag):

*# Επιλογή των στηλών που θα καθυστερήσουν (αφαιρούμε τις άχρηστες στήλες 'timestamp' και 'label')*

cols\_to\_shift = [col for col in df.columns if (col != 'timestamp' and col != 'label')]

*# Δημιουργία ενός λεξικού που περιέχει τις καθυστερημένες στήλες*

lagged\_data = {f'{col}\_lag\_{i}': df[col].shift(i) for col in cols\_to\_shift for i in range(1, lag + 1)}

*# Δημιουργία ενός νέου DataFrame από το λεξικό με τα καθυστερημένα δεδομένα*

lagged\_df = pd.DataFrame(lagged\_data)

*# Συνένωση του αρχικού DataFrame με το DataFrame που περιέχει τα καθυστερημένα δεδομένα*

df = pd.concat([df, lagged\_df], axis=1)

*# Αντικατάσταση των τιμών NaN που δημιουργήθηκαν από τις καθυστερήσεις με 0*

df.fillna(0, inplace=True)

return df

*# Μονοπάτι για τον φάκελο που περιέχει τα αρχεία CSV*

path = '/content/drive/MyDrive/harth'

**Arificial Neural Network**

Αρχικά, δημιουργούμε 4 λίστες, οι οποίες θα χρησιμοποιηθούν για να αποθηκεύουν τις μετρικές απόδοσης. **ann\_accuracies = []**, **ann\_precisions = []**, **ann\_recalls = []**, **ann\_f1\_scores = [].** Θέλουμε να υλοποιήσουμε το νευρωνικό σε όλα τα αρχεία που υπάρχουν μέσα στον φάκελος harth, οπότε διατρέχουμε τα αρχεία ένα ένα και για κάθε αρχείο εκτελούμε την ίδια διαδικασία. Το αποθηκεύουμε σε ένα dataframe **df** και ύστερα πάνω σε αυτό καλούμε την συνάρτηση **create\_lags(df,50),** οπότε για κάθε χρονική στιγμή θα κοιτάμε και τις 50 προηγούμενες. Στην συνέχεια, θα χωρίσουμε το dataframe σε train και test, το train θα αποτελείται από το 80% των δεδομένων και θα είναι τα αρχικά δεδομένα του **df** και το test θα αποτελείται από το υπόλοιπο 20%. Επιπλέον, θα διαχωρίσουμε τις στήλες που θα χρησιμοποιούμε (x: ανεξάρτητες μεταβλητές, y εξαρτημένες μεταβλητές). Τα χαρακτηριστικά **x\_train, y\_train** θα περιλαμβάνουν όλες τις στήλες εκτός από την timestamp και την label και τα χαρακτηριστικά **x\_test, y\_test** θα αποτελούνται μόνο από την στήλη label. Χρησιμοποιώντας την συνάρτηση **tf.keras.utils.to\_categorical()** κωδικοποιούμε την στήλη label με one-hot encoding. Ύστερα με την συνάρτηση **tf.keras.models.Sequential()** αρχικοποιούμε το νευρωνικό μας δίκτυο. Με την συνάρτηση **add()** κατασεκευάζουμε δύο κρυφά επίπεδα με 50 units το καθένα και ενεργοποίηση ReLU καθώς και το επίπεδο εξόδου με ενεργοποίηση softmax. Για την σύνθεση του νευρωνικού χρησιμοποιούμε την συνάρτηση **compile(),** η οποία δέχεται σαν όρισμα τον αλγόριθμος βελτιστοποίησης Adam optimizer, την συνάρτηση απώλειας categorical cρossentropy και την μετρική accuracy. Στην συνέχεια κάνουμε fit το μοντέλο με τη συνάρτηση **fit()**, η οποία δέχεται σαν όρισμα τα δεδομένα εκπαίδευσης, το batch\_size, τα epochs και το verbose. Επιπλέον, πραγματοποιούμε τις προβλέψεις με την **predict()** και υπολογίζουμε τις μετρικές. Αποθηκεύουμε, εκτυπώνουμε τις μετρικές και υπολογίζουμε confusion matrix με την συνάρτηση **confusion\_matrix()** και το εμφανίζουμε ως heatmap. Τέλος, υπολογίζουμε και εκτυπώνουμε τις μέσες τιμές για όλες τις μετρικές για όλους του συμμετέχοντες.

import tensorflow as tf

*# Λίστες για την αποθήκευση των μετρικών από κάθε αρχείο*

ann\_accuracies = []

ann\_precisions = []

ann\_recalls = []

ann\_f1\_scores = []

*# Διατρέχουμε τα αρχεία στον φάκελο*

for filename in os.listdir(path):

if filename.endswith(".csv"):

*# Φορτώνουμε το CSV αρχείο*

df = pd.read\_csv(os.path.join(path, filename))

*# Δημιουργούμε τα lags*

df = create\_lags(df, 50)

*# Διαχωρίζουμε σε train και test sets*

train\_size = int(0.8 \* len(df))

train\_df = df.iloc[:train\_size]

test\_df = df.iloc[train\_size:]

*# Διαχωρισμός features (X) και labels (y) για το train και test set*

X\_train = train\_df.drop(['timestamp', 'label'], axis=1)

y\_train = train\_df['label']

X\_test = test\_df.drop(['timestamp', 'label'], axis=1)

y\_test = test\_df['label']

*# Κωδικοποιούμε τα labels σε κατηγορίες one-hot encoding*

y\_train = tf.keras.utils.to\_categorical(y\_train)

y\_test = tf.keras.utils.to\_categorical(y\_test)

*# Αρχικοποιούμε το Artificial Neural Network*

ann = tf.keras.models.Sequential()

*# Προσθέτουμε το πρώτο κρυφό layer. Χρησιμοποιούμε 50 νευρώνες και την ενεργοποίηση ReLU. Το input\_shape ορίζει την είσοδο, που είναι το training set*

ann.add(tf.keras.layers.Dense(units=50, activation="relu", input\_shape=(X\_train.shape[1],)))

*# Προσθέτουμε το δεύτερο κρυφό layer. Προσθέτουμε ακόμα 50 νευρώνες και την ενεργοποίηση ReLU*

ann.add(tf.keras.layers.Dense(units=50, activation="relu"))

*# Προσθέτουμε το layer εξόδου που έχει τόσους νευρώνες όσες και οι κατηγορίες που έχουμε στο y\_train. Η ενεργοποίηση softmax χρησιμοποιείται για να διασφαλίσουμε ότι οι τιμές εξόδου είναι πιθανότητες που αθροίζουν σε 1*

ann.add(tf.keras.layers.Dense(units=y\_train.shape[1], activation="softmax"))

*# Εφαρμόζουμε το ANN με optimizer "adam" και loss function "categorical\_crossentropy" καθώς έχουμε κατηγορίες που είναι σε μορφή one-hot encoding*

ann.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=['accuracy'])

*# Εκπαιδεύουμε το μοντέλο με τα training data έχοντας batch size 32 και 50 εποχές εκπαίδευσης*

ann.fit(X\_train, y\_train, batch\_size=32, epochs=50, verbose=0)

*# Προβλέπουμε τις δραστηριότητες του test set*

y\_pred = ann.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

y\_test\_classes = np.argmax(y\_test, axis=1)

*# Υπολογίζουμε τις μετρικές*

accuracy = accuracy\_score(y\_test\_classes, y\_pred\_classes)

precision = precision\_score(y\_test\_classes, y\_pred\_classes, average='weighted')

recall = recall\_score(y\_test\_classes, y\_pred\_classes, average='weighted')

f1 = f1\_score(y\_test\_classes, y\_pred\_classes, average='weighted')

*# Προσθήκη των μετρικών στις λίστες*

ann\_accuracies.append(accuracy)

ann\_precisions.append(precision)

ann\_recalls.append(recall)

ann\_f1\_scores.append(f1)

*# Εκτύπωση των μετρικών για κάθε συμμετέχοντα*

base\_name = os.path.splitext(filename)[0] *# Χωρίς κατάληξη*

print(f"Metrics for Participant {base\_name}:")

print(f" Accuracy: {accuracy}")

print(f" Precision: {precision}")

print(f" Recall: {recall}")

print(f" F1-Score: {f1}")

*# Υπολογισμός και εμφάνιση του confusion matrix*

unique\_labels = sorted(df['label'].unique())

cm = confusion\_matrix(y\_test\_classes, y\_pred\_classes)

plt.figure(figsize=(10, 7))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=unique\_labels, yticklabels=unique\_labels)

plt.title(f'Confusion Matrix for Participant {base\_name}')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

*# Υπολογισμός της μέσης τιμής για κάθε μετρική από όλους τους συμμετέχοντες*

ann\_mean\_accuracy = np.mean(ann\_accuracies)

ann\_mean\_precision = np.mean(ann\_precisions)

ann\_mean\_recall = np.mean(ann\_recalls)

ann\_mean\_f1 = np.mean(ann\_f1\_scores)

print("Mean Metrics of All Participants:")

print(f" Mean Accuracy: {ann\_mean\_accuracy}")

print(f" Mean Precision: {ann\_mean\_precision}")

print(f" Mean Recall: {ann\_mean\_recall}")

print(f" Mean F1-Score: {ann\_mean\_f1}")

1. **Ερώτημα 3ο**

**4.1 Περιγραφή**

**Κανονικοίηση**

Επιλέξαμε να χρησιμοποιήσουμε κανονικοποίηση για τα δεδομένα μας, καθώς βοηθά στην σταθεροποίηση της διαδικασίας εκπαίδευσης και τα χαρακτηριστικά κυμαίνονται στην ίδια κλίμακα τιμών, οπότε αποτρέπει ένα χαρακτηριστικό να κυριαρχεί από τα υπόλοιπα λόγω διαφορετικής κλίμακας. Ο μετασχηματιστής που εφαρμόζουμε είναι ο **StandarScaler** της βιβλιοθήκης transformer, ο οποίος θα μετασχηματίζει τα δεδομένα ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1.

**Elbow Method**

Η elbow method είναι μια μέθοδος που χρησιμοποιείται για την εύρεση του βέλτιστου αριθμού συστάδων. Πιο συγκεκριμένα υπολογίζει το σύνολο των τετραγωνικών αποστάσεων των σημείων από τα κοντινότερα κέντρα συστάδων για ένα πλήθος τιμών. Η γραφική παράσταση απεικονίζει της διακυμάνσεις για τα διάφορα k και ο βέλτιστος αριθμός συστάδων θα είναι το σημείο «αγκώνας», όπου ο ρυθμός μείωσης της διακύμανσης εξισώνεται απότομα.

**K-Means**

Ο K-Means είναι διαμεριστικός αλγόριθμος και στόχος του είναι να να βρει την κατάλληλη συσταδοποίηση που θα ελαχιστοποιεί το αποτέλεσμα της συνάρτησης του αθροίσματος των τετραγώνων των σφαλμάτων (SSE) από τα πλησιέστρα κέντρα των συστάδων χρησιμοποιώντας μια άπληστη τεχνική. Επίσης, ο συγκεκριμένος αλγόριθμος παράγει συστάδες με κυρτό σχήμα και ο συνολικός χρόνος εκτέλεσης που απαιτεί είναι O(tnkd), όπου t: οι επαναλήψεις που εκτελούνται, n: τα σημεία που έχουμε, k: οι συστάδες, d: οι d πράξεις που απαιτούνται στις d διαστάσεις. Ο αλγόριθμος λειτουργεί επαναληπτικά αρχικά ορίζει με τυχαίο τρόπο k σημεία στο χώρο των δεδομένων και στην συνέχεια αντιστοιχίζει τα δεδομένα στις συστάδες (τα τοποθετεί στην συστάδα που απέχει την μικρότερη απόσταση από το κέντρο βάρους της) και ενημερώνει τα κέντρα βάρους, . Στη συνέχεια υπολογίζονται οι νέες μέσες τιμές για κάθε συστάδα και η διαδικασία επαναλαμβάνεται μέχρι να βρούμε τα τοπικά ελάχιστα.

**Gaussian Mixture Model**

Τo Gaussian Mixture Model είναι ένα πιθανοτικό μοντέλο που χρησιμοποιείται για την ομαδοποίηση και την εκτίμηση πυκνότητας. Το GMM αποδίδει πιθανότητες στα σημεία δεδομένων, επιτρέποντάς τους να ανήκουν σε πολλές συστάδες ταυτόχρονα και υποθέτει ότι τα δεδομένα παράγονται από πολλές γκαουσιανές συνιστώσες καθεμία από τις οποίες αντιπροσωπεύει μια συστάδα. Επίσης, το μοντέλο υποθέτει ότι υπάρχει ένας ορισμένος αριθμός συνιστωσών, όπου κάθε συνιστώσα είναι μια κατανομή gauss. Άρα, ομαδοποιεί τα δεδομένα που ανήκουν σε μια μόνο γκαουσιανή συνιστώσα.

**4.1 Υλοποίηση**

**Κ-Means**

Για την υλοποιηση του k-means ξεκινάμε ορίζοντας το path που βρίσκονται τα αρχεία και δημιιουργώντας 3 λίστες η **all\_data[]** που θα χρησιμοποιηθεί για την αποθήκευση των δεδομένων από όλα τα αρχεία, η **file\_names[]** θα χρησιμοποιηθεί για την αποθήκευση των ονομάτων των αρχείων και η **mean\_value\_list[]** που θα χρησιμοποιηθεί για την αποθήκευση των μέσων τιμών κάθε αρχείου. Διατρέχουμε όλα τα αρχεία στο φάκελο και για καθένα από αυτά αποθηκεύουμε τα δεδομένα σε ένα dataframe **df** και προσθέτουμε στην λίστα **file\_names** το όνομα του αρχείου. Στην συνέχεια, εξάγουμε τα δεδομένα από τις στήλες 'back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z', τα προσθέτουμε σε ένα dataframe **Χ** και στην συνέχεια στην λίστα **all\_data\_append**. Για κάθε στήλη που επιλέξαμε υπολογίζουμε τις μέσες τιμές με την συνάρτηση **mean()** και τις αποθηκεύουμε στη λίστα **mean\_values\_list**. Όταν το πρόγραμμα διατρέξει όλα τα αρχεία στον φάκελο ενώνει όλα τα δεδομένα από τα αρχεία κάθετα, με την συνάρτηση **concatenate(),** σε έναν ενιαίο πίνακα **all\_data\_comdined**. Ύστερα, με την μέθοδο **fit\_transform** θα υπολογίσουμε την μέση τιμή και την τυπική απόκλιση των δεδομένων και στην συνέχεια θα κανονικοποιήσουμε τα δεδομένα μας, χρησιμοποιώντας τον μετασχηματιστή **StandarScaler** πάνω στα υπολογισμένα στατιστικά μεγέθη και θα τα αποθηκεύσουμε στην λίστα **all\_data\_scaled**. Πριν εφαρμόσουμε τον k-means επιλέξαμε να χρησιμοποιήσουμε την elbow method, ώστε να βρούμε τον κατάλληλο αριθμό clusters για τα δεδομένα που έχουμε. Οπότε υπολογίζουμε το SSE για διφορετικό πλήθος συστάδων(από 1 εως 9). Για κάθε διαφορετικό αριθμό συστάδων εκπαιδεύει το μοντέλο **KMeans** και αποθηκεύει το **insertia** (μετρά πόσο καλά τα δεδομένα ομαδοποιούνται γύρω από τα κέντρα των συστάδων τους) στη λίστα **sse**.Τέλος, σχεδιάζουμε την καμπύλη SSE.

import os

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

*# Ορισμός του φακέλου που περιέχει τα αρχεία CSV*

path = '/content/drive/MyDrive/harth'

*# Λίστες για την αποθήκευση των δεδομένων από όλα τα αρχεία*

all\_data = []

*# Λίστα για τα ονόματα των αρχείων*

file\_names = []

*# Λίστες για τις μέσες τιμές από κάθε αρχείο*

mean\_values\_list = []

*# Διάβασμα των αρχείων στο φάκελο*

for filename in os.listdir(path):

if filename.endswith(".csv"):

*# Διάβασμα του CSV αρχείου*

df = pd.read\_csv(os.path.join(path, filename))

file\_names.append(filename) *# Κρατάμε το όνομα του αρχείου για κάθε δείγμα*

*# Επιλογή των συγκεκριμένων χαρακτηριστικών*

columns = ['back\_x', 'back\_y', 'back\_z', 'thigh\_x', 'thigh\_y', 'thigh\_z']

X = df[columns]

*# Αποθήκευση των δεδομένων στη λίστα*

all\_data.append(X)

*# Υπολογισμός μέσης τιμής κάθε στήλης*

mean\_values = X.mean().values

*# Προσθήκη των μέσων τιμών στη λίστα*

mean\_values\_list.append(mean\_values)

*# Συνδυασμός όλων των δεδομένων σε ένα συνολικό πίνακα*

all\_data\_combined = np.concatenate(all\_data, axis=0)

*# Κανονικοποίηση των πλήρων δεδομένων*

scaler = StandardScaler()

all\_data\_scaled = scaler.fit\_transform(all\_data\_combined)

*# Υπολογισμός του SSE για διαφορετικούς αριθμούς συστάδων*

sse = []

for k in range(1, 10):

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(all\_data\_scaled)

sse.append(kmeans.inertia\_)

*# Σχεδίαση της καμπύλης SSE*

plt.plot(range(1, 10), sse, marker='o', color='blue')

plt.xlabel('Number of clusters')

plt.ylabel('SSE')

plt.title('Elbow Method for Optimal k')

plt.show()

Από την παραπάνω γραφική παρατηρήσαμε ότι το κατάλληλο πλήθος συστάδων είναι το 3, οπότε το αποθηκεύουμε στην μεταβλητή **optimal**. Ξεκινάμε την εφαρμογή του αλγορίθμου KMeans, αρχικοποιούμε τον αλγόριθμο kmeans με 3 συστάδες και ορίζουμε και μια σταθερή τιμή για το **random\_state.** Στην συνέχεια εκπαιδεύουμε τον αλγόριθμο με τα κανονικοποιημένα δεδομένα χρησιμοποιώντας την συνάρτηση **fit()**. Όταν τελειώσει η εκπαίδευση εκτυπώνουμε τα κέντρα των συστάδων (είναι οι μέσες τιμές των δεδομένων που ανήκουν σε κάθε συστάδα). Ύστερα, δημιουργούμε ένα dataframe **mean\_values\_df** το οποίο θα περιέχει τα ονόματα των αρχείων ως δείκτες και τις μέσες τιμές των μετρήσεων που είχαμε υπολογίσει παραπάνω ως δεδομένα. Θα εφαρμόσουμε κανονικοποίηση και στις μέσες τιμές και με βάση αυτές τις τιμές θα προβλέψουμε με την συνάρτηση **predict()** σε ποια συστάδα ανήκει κάθε αρχείο. Θα κατασεκυάσουμε το **clusters\_df,** το οποίο θα περιέχει τα όνομα του αρχείου και την συστάδα που έχει τοποθετηθεί. Τέλος, ομαδοποιούμε με την συνάρτηση **groupby()** τα αρχεία ανά συστάδα και εκτυπώνουμε τα ονόματα των αρχείων κάτω από την συστάδα στην οποία ανήκουν.

*# Επιλογή κατάλληλου πληθους συστάδων*

optimal = 3

*# Εφαρμογή του αλγορίθμου k-means*

kmeans = KMeans(n\_clusters=optimal, random\_state=42) *# Ορισμός αριθμού συστάδων και τυχαίας κατάστασης για επαναληψιμότητα*

kmeans.fit(all\_data\_scaled)

*# Εκτύπωση των κέντρων των συστάδων (clusters)*

print("Cluster Centers:")

for center in kmeans.cluster\_centers\_:

print(center)

*# Δημιουργία DataFrame με τα ονόματα των αρχείων ως δείκτες και τις μέσες τιμές ως δεδομένα*

mean\_values\_df = pd.DataFrame(mean\_values\_list, columns=columns, index=file\_names)

*# Κανονικοποίηση των μέσων τιμών*

scaler = StandardScaler()

mean\_values\_scaled = scaler.fit\_transform(mean\_values\_df)

*# Προβλέψεις για τις συστάδες των μέσων τιμών*

cluster\_assignments = kmeans.predict(mean\_values\_scaled)

*# Δημιουργία DataFrame με τα αρχεία και τις αντίστοιχες συστάδες*

clusters\_df = pd.DataFrame({'filename': file\_names, 'cluster': cluster\_assignments})

*# Ομαδοποίηση και εκτύπωση των αρχείων ανά συστάδα*

grouped = clusters\_df.groupby('cluster')['filename'].apply(list)

print("\nPlacement of Participants in Clusters")

for cluster, files in grouped.items():

print(f"Cluster {cluster}:")

for file in files:

*# Χωρίς κατάληξη*

base\_name = os.path.splitext(file)[0]

print(f" {base\_name}")

**Gaussian Mixture**

Όπως και για τον K-Means έτσι και για το Gussian Mixture πριν την υλοποίησή του επιλέξαμε να χρησιμοποιήσουμε την elbow method, ώστε να βρούμε τον κατάλληλο αριθμό clusters για τα δεδομένα που έχουμε. Ξεκινάμε ορίζοντας μια λίστα **insertia[]**, η οποία θα χρησιμοποιηθεί για την αποθήκευση των τιμών του BIC (μέτρο αξιολόγησης) για κάθε αριθμό συστάδων που δοκιμάζονται. Στη συνέχεια, για διαφορετικό πλήθος συστάδων θα εφαρμόσουμε την συνάρτηση **GaussianMixture()** και θα εκπαιδεύσουμε το μοντέλο πάνω στα δεδομένα που υπάρχουν στην λίστα **all\_data\_scaled**. Επιπλέον, θα ποθηκεύσουμε στη λίστα **insertia** τις τιμές που θα προκύψουν χρησιμοποιώντας την συνάρτηση **bic**. Τέλος, θα απεικονίσουμε γραφικά τις τιμές του bic για κάθε πλήθος συστάδων.

from sklearn.mixture import GaussianMixture

from collections import Counter

inertia = [] *# Λίστα για την αποθήκευση των inertia για κάθε αριθμό συστάδων*

*# Δοκιμάζουμε διαφορετικούς αριθμούς συστάδων*

for k in range(1, 10):

gmm = GaussianMixture(n\_components=k, random\_state=42)

gmm.fit(all\_data\_scaled)

inertia.append(gmm.bic(all\_data\_scaled)) *# Χρησιμοποιούμε το Bayesian Information Criterion (BIC) ως μέτρο αξιολόγησης*

*# Οπτικοποίηση του Elbow Method*

plt.plot(range(1, 10), inertia, marker='o', color='blue')

plt.xlabel('Number of Clusters')

plt.ylabel('BIC')

plt.title('Elbow Method for Optimal k')

plt.xticks(range(1, 10))

plt.grid(True)

plt.show()

Από την παραπάνω γραφική παρατηρήσαμε ότι το κατάλληλο πλήθος συστάδων είναι το 3, οπότε το αποθηκεύουμε στην μεταβλητή **optimal**. Ξεκινάμε την εφαρμογή του αλγορίθμου Gaussian Mixture καλώντας την συνάρτηση **GaussianMixture** και δίνοντας σαν όρισμα 3 συστάδες και μια σταθερή τιμή για το **random\_state**. Στην συνέχεια, εκπαιδεύουμε τον αλγόριθμο με τα κανονικοποιημένα δεδομένα χρησιμοποιώντας την συνάρτηση **fit()** και εκτυπώνουμε τα κέντρα των συστάδων (είναι οι μέσες τιμές των δεδομένων που ανήκουν σε κάθε συστάδα). Επίσης, δημιουργούμε μια λίστα **mean\_values\_list** με τις μέσες τιμές για κάθε συμμετέχοντα και την αποθηκεύουμε και στο dataframe **mean\_values\_df** μαζί με τα ονόματα των αρχείων ως δείκτες. Ύστερα, εφαρμόζουμε κανονικοποίηση στις μέσες τιμές και με βάση αυτές θα υπολογίσουμε την απόσταση κάθε συμμετέχοντα από τα κεντροειδή χρησιμοποιώντας την συνάρτηση **np.linalg.norm().** Επιπλέον, με την συνάρτηση **np.argmin()** θα αντιστοιχίσουμε κάθε συμμετέχοντα στην συστάδα με την μικρότερη απόσταση και τα αποτελέσματα θα τα αποθηκεύσουμε στη μεταβλητή **cluster\_assignments**. Ακόμη, θα κατασκευάσουμε το **clusters\_df**, το οποίο θα περιέχει το όνομα του αρχείου και την συστάδα που έχει τοποθετηθεί. Τέλος, ομαδοποιούμε με την συνάρτηση **groupby()** τα αρχεία ανά συστάδα, εκτυπώνουμε τα ονόματα των αρχείων κάτω από την συστάδα στην οποία ανήκουν και απεικονίζουμε γραφικά την θέση των συμμετεχόντων ανά συστάδα.

*# Επιλογή κατάλληλου πληθους συστάδων*

optimal = 3

*# Εφαρμογή του Gaussian Mixture Model*

gmm = GaussianMixture(n\_components=optimal, random\_state=42)

gmm.fit(all\_data\_scaled)

*# Εκτύπωση των κεντροειδών των συστάδων*

print("Cluster Centers (GMM):")

for i, center in enumerate(gmm.means\_):

print(f"Cluster {i + 1}: {center}")

*# Δημιουργία DataFrame με τις μέσες τιμές για κάθε συμμετέχοντα*

mean\_values\_list = [np.mean(data, axis=0) for data in all\_data]

mean\_values\_df = pd.DataFrame(mean\_values\_list, columns=columns, index=[os.path.splitext(f)[0] for f in file\_names])

*# Κανονικοποίηση των μέσων τιμών*

scaler = StandardScaler()

mean\_values\_scaled = scaler.fit\_transform(mean\_values\_df)

*# Υπολογισμός αποστάσεων από τα κεντροειδή*

distances = np.linalg.norm(mean\_values\_scaled[:, np.newaxis] - gmm.means\_, axis=2)

*# Αντιστοίχιση κάθε συμμετέχοντα στη συστάδα με την ελάχιστη απόσταση*

cluster\_assignments = np.argmin(distances, axis=1)

*# Δημιουργία DataFrame με τα αρχεία και τις αντίστοιχες συστάδες*

clusters\_df = pd.DataFrame({'filename': mean\_values\_df.index, 'cluster': cluster\_assignments})

*# Ομαδοποίηση και εκτύπωση των αρχείων ανά συστάδα*

grouped = clusters\_df.groupby('cluster')['filename'].apply(list)

print("\nPlacement of Participants in Clusters (GMM):")

for cluster, files in grouped.items():

print(f"Cluster {cluster}:")

for file in files:

print(f" {file}")

*# Οπτικοποίηση των συστάδων και των σημείων των συμμετεχόντων*

plt.figure(figsize=(12, 8))

*# Δημιουργία χρωματικής παλέτας για τις συστάδες*

unique\_labels = set(cluster\_assignments)

colors = plt.cm.get\_cmap('tab10', len(unique\_labels)).colors

*# Προσθήκη των σημείων στις συντεταγμένες ανά συστάδα*

for i, assignment in enumerate(cluster\_assignments):

cluster = assignment

x = cluster + np.random.rand() \* 0.1 - 0.05 *# Μικρή τυχαία μετατόπιση*

y = np.random.rand() \* 0.1 - 0.05 *# Μικρή τυχαία μετατόπιση*

plt.scatter(x, y, s=100, color=colors[cluster % len(colors)], label=f'Cluster {cluster + 1}' if i == 0 else "")

plt.text(x + 0.02, y, clusters\_df['filename'][i], fontsize=9, ha='left') *# Σταθερή μετατόπιση για την ετικέτα*

*# Προσθήκη υπομνήματος*

for cluster in unique\_labels:

plt.scatter([], [], color=colors[cluster % len(colors)], label=f'Cluster {cluster + 1}')

plt.title('Clustering of Participants Based on Activities (GMM)')

plt.axis('off')

plt.show()

*# Υπολογισμός του Silhouette Score*

silhouette\_avg = silhouette\_score(mean\_values\_scaled, cluster\_assignments)

print(f"\nSilhouette Score for GMM: {silhouette\_avg})”