Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

Εργαστηριακή Άσκηση
Εαρινό Εξάμηνο 2023-2024

Διδάσκοντες:
Καθηγητής Β. Μεγαλοοικονόμου,
Αναπληρωτής Καθηγητής Χ. Μακρής

Στεργιόπουλος Γεώργιος

ΑΜ: 1083861

Εξάμηνο: 8ο

Τμήμα: ΗΜΤΥ

Κατεύθυνση: Υπολογιστές: Λογισμικό και υλικό

Περιεχόμενα

Εισαγωγή

Η υλοποιητική αυτή εργασία, εκπονήθηκε στα πλαίσια του μαθήματος «Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης». Στην εργασία αυτή στόχος είναι η αξιοποίηση ενός παρεχόμενου συνόλου δεδομένων, για την εξοικείωση με το αντικείμενο του μαθήματος.

Συγκεκριμένα, μας δόθηκε το σύνολο δεδομένων Harth, το οποίο αποτελείται από 6461328 δεδομένα, που έχουν συλλεχθεί από πείραμα που διεξήχθη. Στο πείραμα αυτό 22 συμμετέχοντες, εφοπλισμένοι με 2 αισθητήρες κίνησης ο καθένας, σε διαφορετικό σημείο του σώματος τους, καταγράφηκαν να εκτελούν διάφορες δραστηριότητες μέσα σε ένα επιβλεπόμενο χώρο για συγκεκριμένο χρονικό διάστημα. Τα δεδομένα είναι εφοπλισμένα με το κατάλληλο label, για κάθε χρονική στιγμή που δείχνει την δραστηριότητα του χρήστη.

Εμείς, όπως θα δείτε στην παρακάτω αναφορά, σε πρώτο στάδιο καλούμαστε να κάνουμε την κατάλληλη προ-επεξεργασία και στατιστική ανάλυση των δεδομένων αυτών, με σκοπό να καταλάβουμε τα δεδομένα μας, άλλα και να τα φέρουμε στην καταλληλότερη μορφή για τα επόμενα βήματα.

Έπειτα, μας ζητείται να εκπαιδεύσουμε 3 ταξινομητές, έναν που βασίζεται σε Neural Networks, έναν σε Random Forest και έναν σε Bayesian Networks για να κάνουμε Classification.

Τέλος, θα προσπαθήσουμε μέσω 2 αλγορίθμων συσταδοποίησης να εφαρμόσουμε τεχνικές unsupervised learning, για να χωρίσουμε τους χρήστες σε συστάδες με βάση την δραστηριότητα τους.

Περιβάλλον Υλοποίησης

Hardware

Η εργασία αυτή υλοποιήθηκε σε υπολογιστή, με λειτουργικό σύστημα Windows 11, και τα παρακάτω χαρακτηριστικά.

Τα παραπάνω χαρακτηριστικά παρατίθενται ως αναφορά, γιατί σχετίζονται άμεσα με τους χρόνους εκτέλεσης του κώδικα της εργασίας. Οποιοσδήποτε χρόνος αναγράφεται σε σχόλια στον κώδικα, έχει καταγραφεί από αυτό το σύστημα.

Εργαλεία και βιβλιοθήκες

VSCode και GitHub

github.com/gdstergiopoulos/dataMining_project

Ο κώδικας γράφτηκε εξ ολοκλήρου στο VSCode, και χρησιμοποιήθηκε Github για την διαχείριση της εργασίας και των εκδόσεων.

Στο Github Repo, βρίσκεται ο κώδικας από το τελικό παραδοτέο μου (up1083861.ipynb) αλλα και ένας φάκελος old_tries οπού περιέχει δοκιμές και διάφορα test που οδήγησαν στην ολοκλήρωση της εργασίας μου.

Στο github, δεν έχει γίνει commit η ΒΔ, συνεπώς αν θέλει κάποιος να κατεβάσει και να εκτελέσει τον κώδικα από εκεί πρέπει να δημιουργήσει μόνος του την ΒΔ, όπως περιγράφεται παρακάτω στο “DB Browser for SQLite”.

Jupyter Notebooks

Όλη η εργασία έχει δημιουργηθεί σε Jupyter Notebooks (.ipynb), καθώς αυτά μας δίνουν την δυνατότητα, της εκτέλεσης κομματιών κώδικα μεμονωμένα. Αυτό είναι ιδιαίτερα χρήσιμο για εργασίες που απαιτούν αρκετή υπολογιστική ισχύ και διαρκούν αρκετό χρόνο. Για παράδειγμα έστω ότι θέλουμε να εκπαιδεύσουμε 3 ταξινομητές, ξεχωριστούς με τα ίδια δεδομένα. Αξιοποιώντας απλά .py αρχεία θα έπρεπε είτε να φορτώνουμε σε κάθε αρχείο ξεχωριστά τα data ή κάθε φορά να εκπαιδεύουμε εκ νέου τους κατηγοριοποιητές, ενώ πραγματοποιήσαμε αλλαγές μόνο σε κάποιον από αυτούς. Το Jupyter notebook, επιλύει αυτά τα ζητήματα μέσω τον cells.

Επιπλέον, επιτρέπει τον συνδυασμό του κώδικα με κείμενο, με ευανάγνωστο τρόπο για αναλυτική επεξήγηση αυτού.

Ως kernel, για εκτέλεση των notebook cells, αξιοποιήσαμε την Python 3.12.0, που τρέχει locally στον υπολογιστή.

DB Browser for SQLite

Το DBMS αυτό αξιοποιήθηκε με σκοπό να μετατρέψουμε τα αρχεία .csv, που κατεβάσαμε σε μία βάση δεδομένων.

Για να γίνει αυτό δημιουργήσαμε μέσω της εφαρμογής αυτής μια νέα ΒΔ, με όνομα “harth.db” και κάναμε import data από τα .csv αρχεία διατηρώντας το όνομα του κάθε αρχείου ως όνομα του πίνακα.

Βιβλιοθήκες Python

Sqlite3

Κατεβάζοντας το Dataset, από το internet, παρατήρησα ότι ήταν σε μορφή 22 αρχείων CSV. Για καθαρά λόγους προτίμησης στην σύνταξη του κώδικα μου, αποφάσισα να δημιουργήσω μια sqlite βάση δεδομένων με 22 πίνακες καθένας από τους οποίους έχει τα δεδομένα και το όνομα από το αντίστοιχο CSV αρχείο.

Προφανώς στον κώδικα συντάχθηκαν αντίστοιχα SQL queries, ανάλογα τις ανάγκες του προγράμματος.

Η SQLite3 είναι μια default βιβλιοθήκη της Python και ως εκ τούτου δεν χρειάζεται κάποια εγκατάσταση. Αρκεί το import αυτής.

```
import sqlite3
```

Pandas

Η βιβλιοθήκη Pandas της Python, αποτέλεσε βασικό εργαλείο για αυτήν την εργασία.

Τα δεδομένα που επέστρεψε η SQLite μετά τα κατάλληλα Queries, μετατράπηκαν σε Pandas Dataframe. Με αυτόν τον τρόπο, είχα την δυνατότητα να εφαρμόσω διάφορες μεθόδους στα δεδομένα, για να τα μελετήσω, να τα επεξεργαστώ και να τα μετατρέψω κάθε φορά στο format που ήθελα για να τα αξιοποιήσω.

Χαρακτηριστικά παραδείγματα μεθόδων ήταν:

- Drop: για να αφαιρέσουμε στήλες που δεν χρειάζονται
- Info: παρέχει πληροφορίες για τον τύπο τον δεδομένων του df
- Describe: υπολογίζει διάφορα μεγέθη για το data frame ανά στήλη, πχ μέση τιμή, τυπική απόκλιση κ.α.
- Rolling: Αυτή η μέθοδος αξιοποιήθηκε για να δημιουργήσουμε ένα rolling window, πάνω στα δεδομένα και να εφαρμόσουμε υπολογισμούς σε αυτά (πχ υπολογισμός mean μέσα από το window)

Οδηγίες Εγκατάστασης:

Σε έναν υπολογιστή με εγκατεστημένη την Python, αρκεί σε ένα terminal, να εκτελέσουμε την εντολή

pip install pandas

Έπειτα η βιβλιοθήκη pandas, μπορεί να γίνει import και να αξιοποιηθεί στους κώδικες μας.

```
import pandas as pd
```

Numpy

Σε κάποια σημεία του κώδικα αξιοποιήθηκε η numpy για μαθηματικούς υπολογισμούς σε arrays.

Οδηγίες Εγκατάστασης:

Σε έναν υπολογιστή με εγκατεστημένη την Python, αρκεί σε ένα terminal, να εκτελέσουμε την εντολή

pip install numpy

Έπειτα η βιβλιοθήκη numpy, μπορεί να γίνει import και να αξιοποιηθεί στους κώδικες μας.

```
import numpy as np
```

Matplotlib

Η βιβλιοθήκη Matplotlib και συγκεκριμένα το module pyplot αυτής, αποτέλεσε το βασικό εργαλείο για δημιουργία γραφικών παραστάσεων. Μέσω αυτής δημιουργήθηκαν τα

περισσότερα διαγράμματα τόσο στο στάδιο της προ-επεξεργασίας, όσο και στα επόμενα ερωτήματα όπου αυτά απαιτήθηκαν.

Οδηγίες Εγκατάστασης:

Σε έναν υπολογιστή με εγκατεστημένη την Python, αρκεί σε ένα terminal, να εκτελέσουμε την εντολή

pip install matplotlib

Έπειτα η βιβλιοθήκη matplotlib, μπορεί να γίνει import και να αξιοποιηθεί στους κώδικες μας.

```
import matplotlib.pyplot as plt
```

Seaborn

Η βιβλιοθήκη Seaborn, η οποία βασίζεται στην Matplotlib, παρέχει κάποιες πιο εξειδικευμένες δυνατότητες αναπαράστασης δεδομένων και των στατιστικών αυτών. Για αυτό το λόγο αξιοποιήθηκε, η συνάρτηση heatmap αυτής. Η heatmap μας έδωσε την δυνατότητα να δούμε τις συσχετίσεις των δεδομένων του συνόλου μας, μεταξύ τους, σε ένα ευανάγνωστο γράφημα.

Οδηγίες Εγκατάστασης:

Σε έναν υπολογιστή με εγκατεστημένη την Python, αρκεί σε ένα terminal, να εκτελέσουμε την εντολή

pip install seaborn

Έπειτα η βιβλιοθήκη seaborn, μπορεί να γίνει import και να αξιοποιηθεί στους κώδικες μας.

```
import seaborn as sns
```

Scikit-learn (sklearn)

Περνάμε τώρα στην βασική βιβλιοθήκη που αξιοποιήθηκε για αυτήν την εργασία, τόσο για το ερώτημα του Classification, όσο και για του Clusterring.

Η sklearn είναι μια, ανοιχτού κώδικα βιβλιοθήκη μηχανικής μάθησης, η οποία παρέχει βασικές δυνατότητες για ανάλυση και εξόρυξη δεδομένων.

Συνεργάζεται, ιδιαίτερα καλά με τις βιβλιοθήκες pandas και numpy, που όπως περιγράψαμε παραπάνω, έχουν αξιοποιηθεί σε αυτή την εργασία.

Model selection:

train test split:

Το train test split function της sklearn.model_selection, μας παρείχε την δυνατότητα να χωρίσουμε τα δεδομένα μας σε train and test.

```
from sklearn.model_selection import train_test_split
```

GridSearchCV

Το GridSearchCV, παρέχει την δυνατότητα να προσπαθήσουμε, να βρούμε τις βέλτιστες παραμέτρους για κάθε μοντέλο που εκπαιδεύουμε για να πετύχουμε την βέλτιστη απόδοση

```
from sklearn.model_selection import GridSearchCV
```

Preprocessing

Standardscaler

Μέσω του Standard-scaler, έγινε κανονικοποιήση των δεδομένων, πράγμα που εξυπηρετεί την απόδοση αρκετών μοντέλων τόσο στην Κατηγοριοποίηση, όσο και στην ταξινόμηση.

```
from sklearn.preprocessing import StandardScaler
```

Neural network:

MLPClassifier

Για την εκπαίδευση του ταξινομητή που βασίζεται σε Νευρωνικό Δίκτυο, επιλέχθηκε ο MLPClassifier. Η κλάση αυτή της sklearn, παρέχει την δυνατότητα να εκπαιδεύσουμε ένα νευρωνικό δίκτυο με όσα hidden layer επιθυμούμε, επιλέγοντας παραμέτρους όπως το Activation Function κλπ.

```
from sklearn.neural_network import MLPClassifier
```

Ensemble:

Random Forest Classifier

Αντίστοιχα για την εκπαίδευση του ταξινομητή που βασίζεται σε δέντρα απόφασης, επιλέχθηκε ο Random Forest Classifier, όπου αποτελεί κλάση του Ensemble module της sklearn. Συνδυάζει δηλαδή τις αποφάσεις αρκετών δέντρων αποφάσεων για να κάνει την ταξινόμηση.

```
from sklearn.ensemble import RandomForestClassifier
```

Naïve bayes:

GaussianNB

Τέλος, για την εκπαίδευση ταξινομητή βασισμένος σε Bayesian Networks , αξιοποιήθηκε η κλάση GaussianNB, του module Naïve Bayes της sklearn.

```
from sklearn.naive_bayes import GaussianNB
```

Cluster:

Kmeans:

Για την διαδικασία της κατηγοριοποίησης, αξιοποιήσαμε την κλάση KMeans του module Cluster, της βιβλιοθήκης sklearn. Μέσω αυτού, καταφέραμε να κάνουμε ομαδοποίηση στα δεδομένα μας.

```
from sklearn.cluster import KMeans
```

Agglomerative Clustering:

Ο δεύτερος αλγόριθμος που επιλέχθηκε ήταν ο Agglomerative Clustering, ο οποίος υλοποιείται στην βιβλιοθήκη sklearn, από την αντίστοιχα ονομαζόμενη κλάση.

```
from sklearn.cluster import AgglomerativeClustering
```

Metrics:

Για την αξιολόγηση των αποτελεσμάτων των ταξινομητών αξιοποιήθηκε το module Metrics.

Accuracy_score, Classification Report:

Η συνάρτηση classification report επιστρέφει μια αναλυτική αναφορά μετρικών μεγεθών που δείχνουν την επίδοση του μοντέλου μας. Πχ δίνει το precision, το recall, f1-score κ.α. Αντίστοιχα το accuracy score δίνει την ακρίβεια της ταξινόμησης στα test data.

Silhouette Score:

Η συνάρτηση silhouette score του module metrics της sklearn, μας βοήθησε στην παραμετροποίηση του KMeans, αλγορίθμου. Μας βοήθησε να προσεγγίσουμε με έναν διαφορετικό τρόπο την παραμετροποίηση του KMeans.

Οδηγίες Εγκατάστασης:

Σε έναν υπολογιστή με εγκατεστημένη την Python, αρκεί σε ένα terminal, να εκτελέσουμε την εντολή

pip install scikit-learn

Έπειτα η scikit-learn είναι διαθέσιμη για import στον κώδικα μας με τον τρόπο που φαίνεται στις διάφορες φωτογραφίες παραπάνω. Χρησιμοποιείτε το ψευδώνυμο sklearn.

Yellowbrick:

Στο 3ο ερώτημα αξιοποιήσαμε και την βιβλιοθήκη yellowbrick και συγκεκριμένα το KElbowVisualizer, το οποίο υλοποιεί την μέθοδο KElbow, για εύρεση της κατάλληλης τιμής της παραμέτρου k.

Οδηγίες Εγκατάστασης:

Σε έναν υπολογιστή με εγκατεστημένη την Python, αρκεί σε ένα terminal, να εκτελέσουμε την εντολή

pip install yellowbrick

Έπειτα η βιβλιοθήκη yellowbrick, μπορεί να γίνει import και να αξιοποιηθεί στους κώδικες μας.

```
from yellowbrick.cluster import KElbowVisualizer
```

Διαδικασία Υλοποίησης

Η διαδικασία υλοποίησης της εργασίας είχε ως εξής.

Αρχικά μελέτησα την ιστοσελίδα και το paper, όπου περιγράφουν τον τρόπο που συλλέχθηκαν καθώς και τι αναπαριστούν τα δεδομένα του dataset.

Έπειτα, και με βάση τις οδηγίες, που μας δόθηκαν στο φροντιστήριο του μαθήματος, ξεκίνησε η υλοποίηση του κώδικα ο οποίος απαντάει στα ερωτήματα της εκφώνησης.

Αφότου όπως ανέφερα παραπάνω, δημιούργησα την βάση δεδομένων και την γέμισα με τα δεδομένα που μας δόθηκαν, προχώρησα στο στάδιο της ανάλυσης αυτών.

Σε αυτό το σημείο, χρησιμοποιήθηκαν συναρτήσεις της Pandas και της Matplotlib, έτσι ώστε να δούμε τις στατιστικές κατανομές των δεδομένων, τους outlier, τις συσχετίσεις τους, την ύπαρξη ή μη NaN values κ.α.

Με βάση τα παραπάνω, στατιστικά στοιχεία και κάποιες άλλες παρατηρήσεις που έγιναν, επεξεργάστηκα κατάλληλα τα Dataframe, για να είναι στην μορφή που απαιτούταν για τα επόμενα ερωτήματα. Για παράδειγμα σε κάποια δεδομένα των tester, υπήρχαν στήλες με indexes, τα οποία ήταν άχρηστα για το σύνολο δεδομένων και δεν προσέφεραν καμία έξτρα πληροφορία. Θα γίνει περεταίρω ανάλυση παρακάτω.

Με την παραπάνω διαδικασία να έχει ολοκληρωθεί, προχώρησα στο ερώτημα 2 και στην εκπαίδευση διάφορων μοντέλων ταξινομητών. Συγκεκριμένα, σε έναν ταξινομητή βασισμένο σε Νευρωνικά Δίκτυα, έναν σε Δέντρα Απόφασης και έναν σε Bayesian Networks.

Εδώ προσέγγισα το ζήτημα με διαφορετικούς τρόπους.

- Εκπαίδευση με τα δεδομένα, αγνοώντας την χρονική τους αλληλουχία

Ξεκινώντας την προσπάθεια μου, αποφάσισα να εκπαιδεύσω τα μοντέλα, περνώντας τους τα δεδομένα, κάνοντας drop το timestamp. Δηλαδή τα δεδομένα ήταν σαν ξεχωριστά μεμονωμένα για το μοντέλο, το οποίο δεν είχε πληροφορία για το τι συνέβαινε πριν ή μετά το συγκεκριμένο sample.

- Εκπαίδευση με τα δεδομένα χρησιμοποιώντας rolling window, οπού υπολογίζει μέσες τιμές

Ως δεύτερη προσέγγιση και θέλοντας να δώσουμε στα μοντέλα πληροφορία για τις μετρήσεις τις προηγούμενες και τις επόμενες χρονικές στιγμές, δημιουργήσαμε ένα rolling window. Αυτό για κάθε δείγμα, λαμβάνοντας υπόψιν του τις 5 στιγμές πριν και 5 μετά, υπολόγιζε έναν μέσο όρο για τις μετρήσεις, ο οποίος μετά πέρναγε ως είσοδος στα μοντέλα. Έτσι με έναν, έμμεσο τρόπο, τα δεδομένα δεν ήταν χρονικά ασυσχέτιστα μεταξύ τους.

- Εκπαίδευση με τα δεδομένα να περιέχουν αμιγώς τις τιμές των αισθητήρων τις προηγούμενες και τις επόμενες χρονικές στιγμές.

Τέλος, προσπάθησα να δημιουργήσω πάλι μέσω ενός rolling window, ένα dataframe το οποίο για κάθε sample, δίνει στο μοντέλο και τα δεδομένα 5 προηγούμενων και 5 επόμενων χρονικών στιγμών. Ωστόσο, η δημιουργία αυτού του data frame, άλλα και η εκπαίδευση των μοντέλων με αυτό ήταν αυξημένης πολυπλοκότητας (υπολογιστικά). Για αυτόν τον λόγο κατέφυγα στο sampling. Πήρα ένα δεδομένο για κάθε 10 δείγματα. Έπειτα από στατιστική ανάλυση, είδα ότι η πληροφορία (μέσες τιμές κλπ.) δεν άλλαζαν πολύ και ως εκ τούτου προχώρησα με αυτό το δειγματοληπτικό data frame.

Σε αυτό το σημείο να σχολιάσουμε, ότι εφαρμόστηκε κανονικοποίηση των δεδομένων, μέσω StantardScaler, άλλα και χωρισμός των data frame σε train και test, μέσω του train test split. Επιπλέον, για κάθε εκπαίδευση μοντέλου και στις 3 προσεγγίσεις υπολογίστηκαν διάφορες μετρικές όπως Accuracy score κλπ.

Σχολιασμός Αποτελεσμάτων

Στο σημείο αυτό θα γίνει σχολιασμός κάποιων αποτελεσμάτων, με ενδεικτικά screenshot από αποτελέσματα του κώδικα. Πολλά από αυτά που θα αναφερθούν ή θα παρουσιαστούν σε μορφή κειμένου ή πινάκων μπορείτε να τα βρείτε στο up1083861.ipynb, όπου σε αρκετά σημεία υπάρχει extra σχολιασμός.

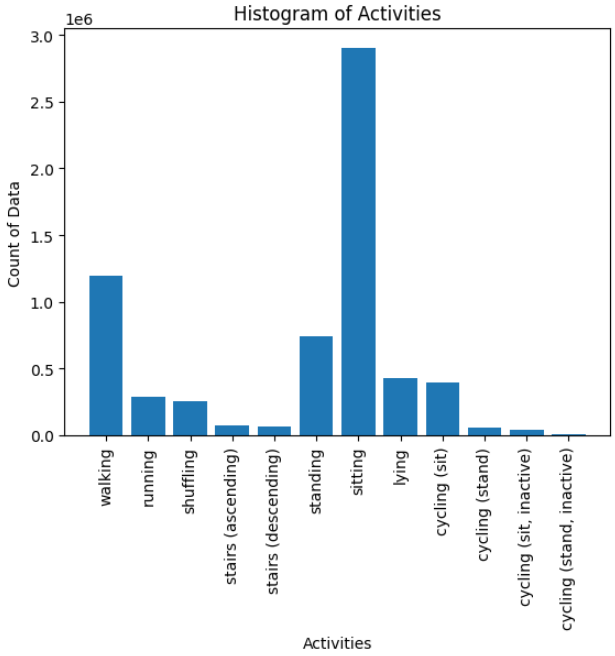
Ερώτημα 1:

Στην πρώτη φάση που κοιτάμε από τι στήλες και δεδομένα αποτελείτε το dataset κάθε tester, βλέπουμε ότι δεν υπάρχουν σε κανέναν null τιμές. Επιπλέον βλέπουμε ότι όλοι έχουν τις εξής στήλες

```
...   Info about tester: S006
      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 408709 entries, 0 to 408708
      Data columns (total 8 columns):
       #   Column     Non-Null Count   Dtype
      ---  ------     --------------   -----
       0   timestamp  408709 non-null  object
       1   back_x     408709 non-null  float64
       2   back_y     408709 non-null  float64
       3   back_z     408709 non-null  float64
       4   thigh_x    408709 non-null  float64
       5   thigh_y    408709 non-null  float64
       6   thigh_z    408709 non-null  float64
       7   label      408709 non-null  int64
```

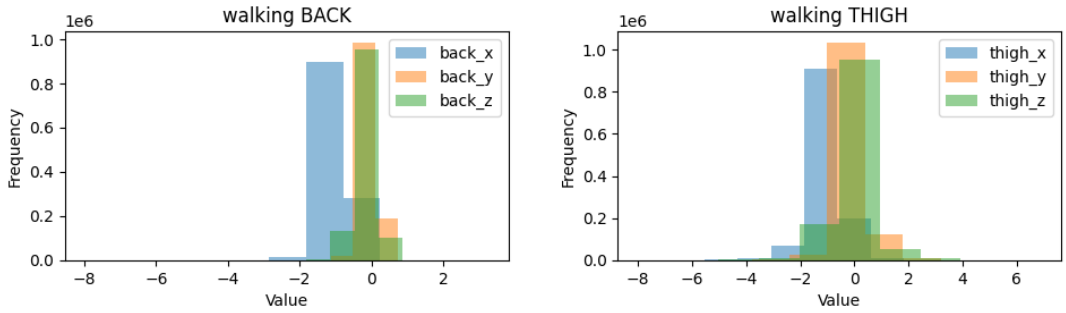
Σε κάποιους tester, παρατηρήσαμε ότι υπήρχαν κάποιες στήλες που περιείχαν indexes, τα οποία δεν μας έδινα extra πληροφορία και έτσι έγιναν drop, με τα δεδομένα να έχουν όλα πλέον την παραπάνω δομή.

Έπειτα αφού ενώσαμε τα δεδομένα όλων των tester, σε ένα ενιαίο dataframe, προχωρήσαμε στην εξαγωγή στατιστικών, όπως πχ πόσα δεδομένα έχουμε ανά δραστηριότητα.



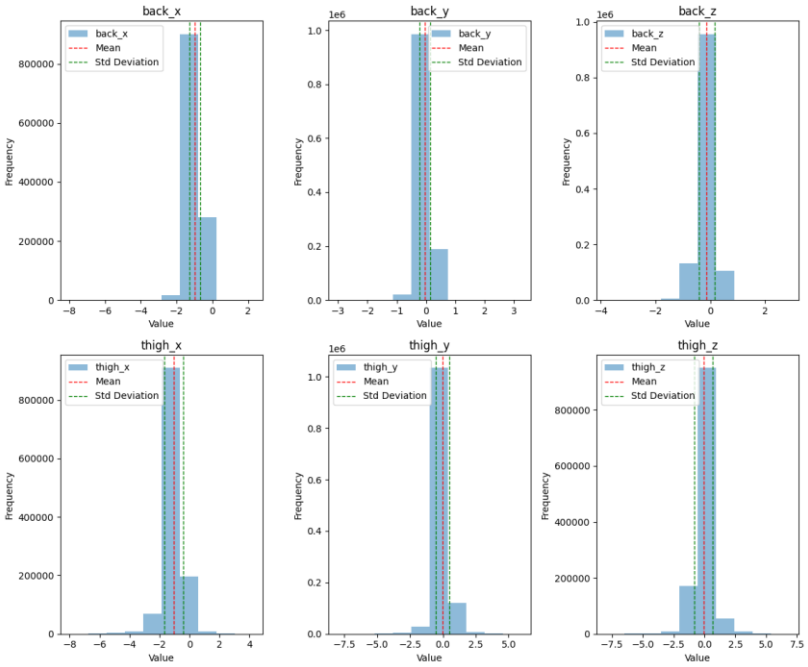
Για κάθε δραστηριότητα, δημιούργησα ένα γράφημα όπου δείχνει σε ιστόγραμμα την κατανομή των δεδομένων, ανά αισθητήρα (back,thigh) και ανά δραστηριότητα.

Ενδεικτικά

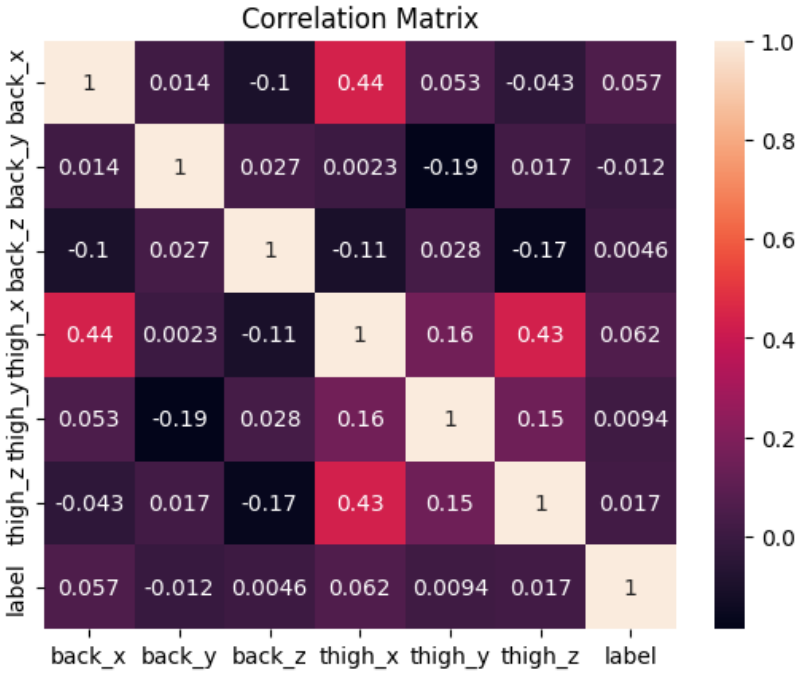


Παρατηρούμε ότι ακολουθούν μια υποτυπώδη κανονική κατανομή. Ωστόσο στο επόμενο στάδιο, δημιουργώ για κάθε δραστηριότητα πιο λεπτομερή γραφήματα, στα οποία αποτυπώνεται κάθε μέτρηση του κάθε αισθητήρα, συνοδευόμενα από κάθετες γραμμές που αποτυπώνουν την μέση τιμή και την τυπική απόκλιση έτσι ώστε να δούμε με μεγαλύτερη λεπτομέρεια κατά πόσο αυτά ακολουθούν κάποια κατανομή.

Ενδεικτικά:

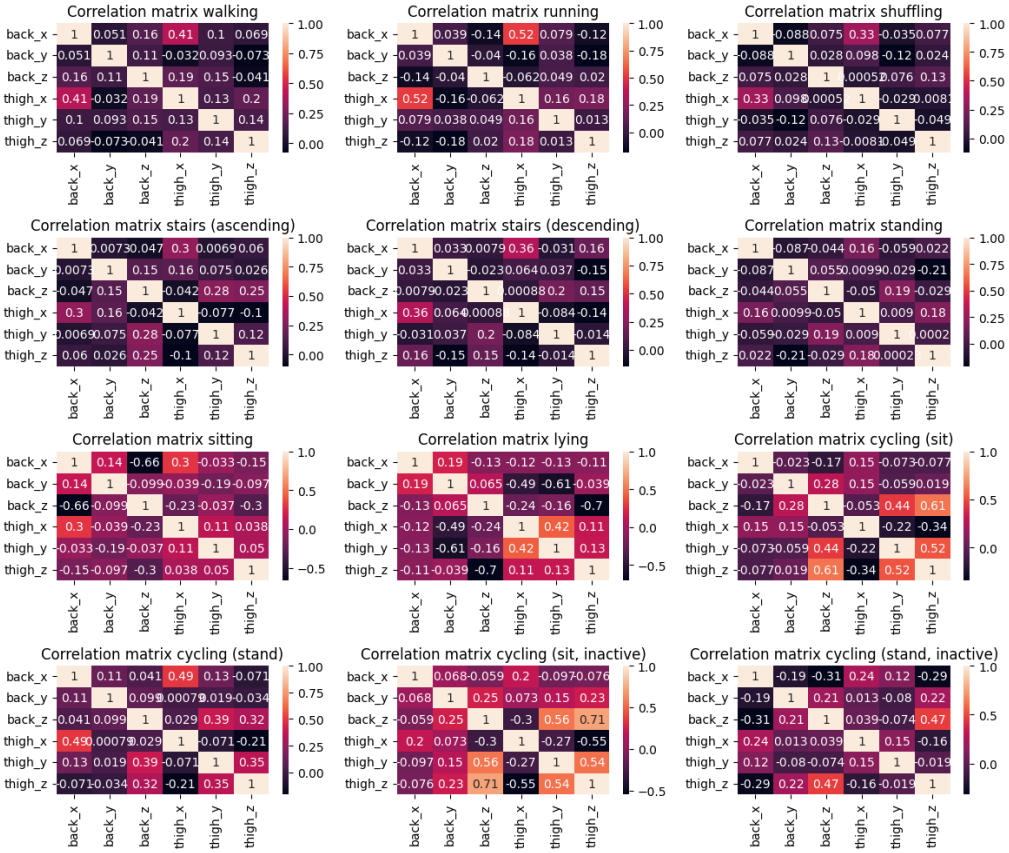


Τέλος, προχώρησα στην δημιουργία πινάκων συσχετίσεων, έτσι ώστε να δούμε πως σχετίζονται μεταξύ τους τα δεδομένα, αλλά και ποια από αυτά έχουν την μεγαλύτερη συσχέτιση με το Label.

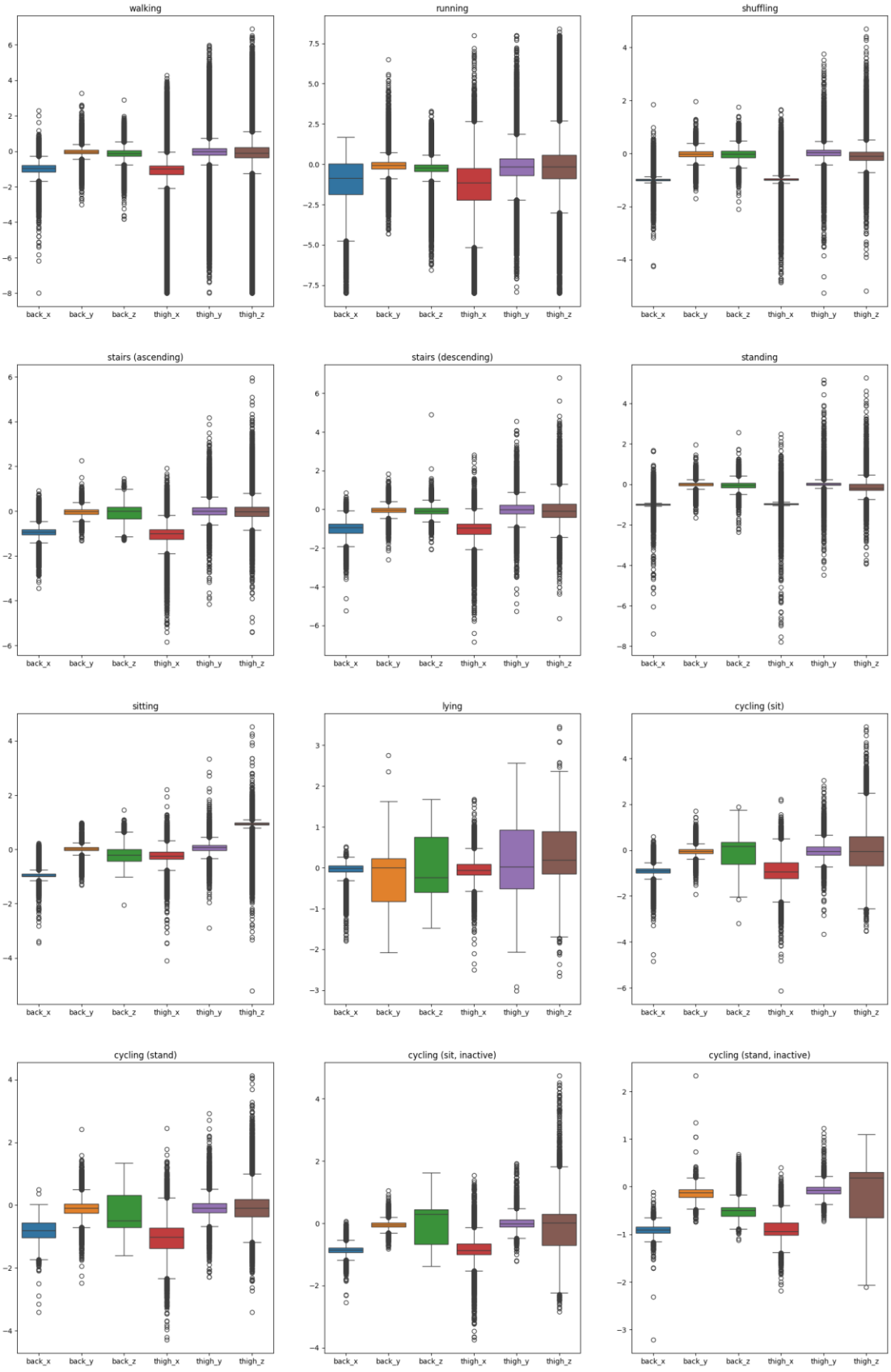


Παρατηρούμε εδώ, έντονη συσχέτιση των μετρήσεων του back_x και thigh_x, αλλά και των thigh_x και thigh_z. Επιπλέον βλέπουμε, ότι συγκριτικά, το back_x είναι αυτό που έχει την μεγαλύτερη συσχέτιση με το label (χωρίς αυτή να είναι απαραίτητα μεγάλη).

Αποβαλόντας το label, προχώρησα στην δημιουργία αντίστοιχων πινάκων συσχέτισης για κάθε δραστηριότητα ξεχωριστά, για να παρατηρήσω λεπτομερώς το πώς οι μετρήσεις κάθε αισθητήρα συσχετίζονται με βάση την κίνηση που εκτελεί εκείνη την στιγμή ο tester.



Τέλος προχώρησα στο να δω και να ερμηνεύσω την ύπαρξη outlier, δημιουργώντας κάποια boxplot ανά δραστηριότητα



Παρατηρώ ότι γενικά υπάρχουν outliers, ιδιαίτερα στις έντονα δυναμικές ασκήσεις όπως στο τρέξιμο κλπ. Επιπλέον παρατηρώ ότι σε αυτές τις έντονες δραστηριότητες έχουμε μεγαλύτερη διαφοροποίηση των δεδομένων, το οποίο είναι αρκετά λογικό καθώς κάθε

tester, μπορεί να περπατάει, να τρέχει, να κάνει ποδήλατο σε διαφορετική ένταση/ρυθμό, κάτι που δεν αποτυπώνεται στα label.

Αντίθετα στατικές δραστηριότητες, όπως το ξάπλωμα, δεν μπορούν να έχουν μεγάλες διαφοροποιήσεις, καθώς είναι μικρότερο το περιθώριο διαφορετικής συμπεριφοράς ανά χρήστη.

Ερώτημα 2:

Η μεθοδολογία που ακολουθήθηκε αναλύθηκε παραπάνω, και ο τρόπος που αυτά υλοποιήθηκαν στην Python φαίνεται στο Jupyter Notebook που συνοδεύει αυτή την αναφορά, συνεπώς εδώ θα περιοριστούμε στον σχολιασμό των αποτελεσμάτων.

Στον παρακάτω πίνακα φαίνεται, ανά προσέγγιση, το accuracy και το training score του κάθε μοντέλου ταξινόμησης. Παρατηρούμε ότι το accuracy αυξάνεται αισθητά από τα μοντέλα της πρώτης, στα μοντέλα της δεύτερης προσέγγισης.

Αυτό είναι αναμενόμενο, καθώς μιλάμε για δεδομένα τα οποία είναι χρονικά συσχετιζόμενα και όχι ανεξάρτητα μεταξύ τους. Οπότε στο πρώτο μοντέλο που αυτή η πληροφορία χάνεται παρατηρούμε χαμηλότερα score στις μετρικές.

Στην Τρίτη προσέγγιση παρατηρούμε μικρή αύξηση του accuracy στον MLP Classifier και μια μικρή, έως μηδαμινή, πτώση στα άλλα 2 μοντέλα. Ωστόσο, αυτό δεν είναι ενδεικτικό καθώς, όπως προ-είπαμε στην 3η προσέγγιση εφαρμόστηκε sampling για μείωση της υπολογιστικής πολυπλοκότητας του προβλήματος.

	1η Προσέγγιση		2η Προσέγγιση		3η Προσέγγιση	
	Train	Test	Train	Test	Train	Test
MLP Classifier	0.8783	0.878	0.8961	0.8955	0.9346	0.927
RF Classifier	0.999	0.9016	0.999	0.942	1	0.9325
GaussianNB	0.7751	0.775	0.8127	0.8124	0.7949	0.7945

Στο Jupyter Notebook, μπορείτε να δείτε αναλυτικότερες μετρικές για κάθε εκπαίδευση μοντέλου καθώς και γραφήματα ανά προσέγγιση.

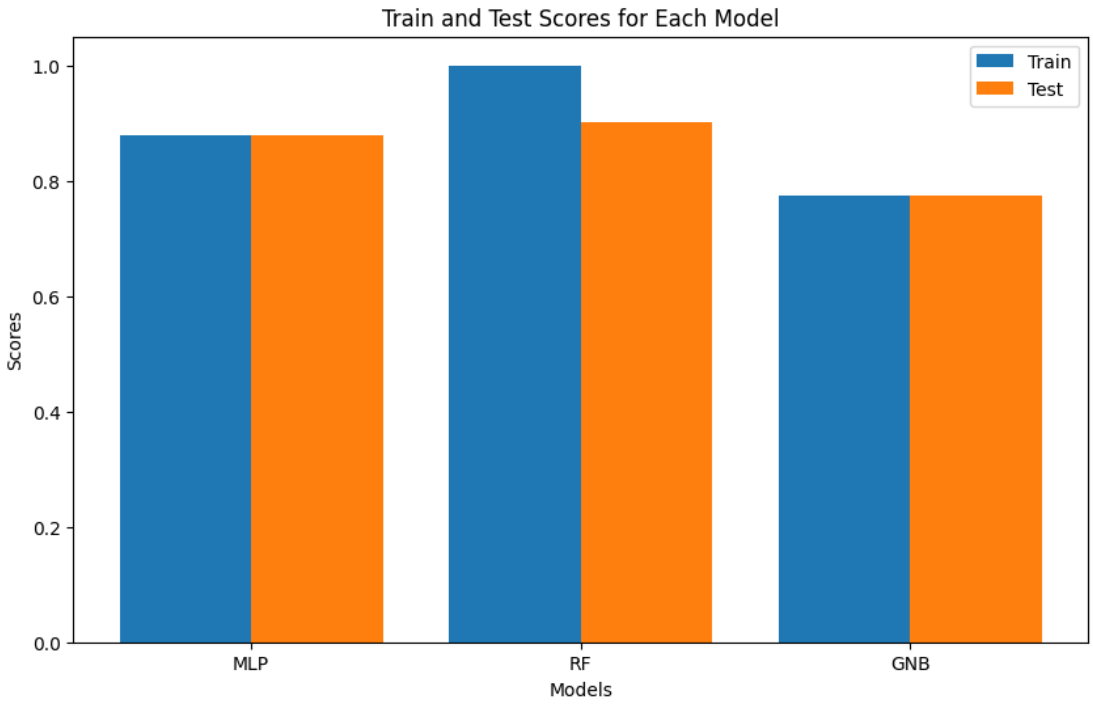
Ενδεικτικά για την πρώτη προσέγγιση, το report του MLP Classifier:

Και τ

```
MLP report:                precision    recall  f1-score   support

                                1       0.76      0.85      0.80    239594
                                2       0.90      0.83      0.86     58550
                                3       0.43      0.17      0.24     51051
                                4       0.33      0.05      0.08     15304
                                5       0.34      0.02      0.04     13441
                                6       0.75      0.89      0.81    148170
                                7       0.99      1.00      0.99    580473
                                8       1.00      1.00      1.00     85900
                               13       0.79      0.85      0.82     78847
                               14       0.59      0.51      0.55     11086
                              130       0.49      0.42      0.45      8275
                              140       0.59      0.27      0.37      1575

                         accuracy                           0.88   1292266
                        macro avg       0.66      0.57      0.59   1292266
                     weighted avg       0.86      0.88      0.86   1292266
MLP accuracy: 0.8780491013460077 MLP training: 0.8782935859542795
```



Τι παρατηρούμε άρα;

Σε κάθε προσέγγιση ο Random Forest Classifier, έχει το υψηλότερο accuracy score στα testing δεδομένα μας, πράγμα που ήταν σχετικά αναμενόμενο καθώς είναι αρκετά δυνατός στο να αντιμετωπίζει δεδομένα με κάποιο θόρυβο αλλά και να ανιχνεύει «σύνθετες» συσχετίσεις. Επιπλέον είναι ανεκτικός στο overfitting και το γεγονός ότι συνδυάζει πολλαπλά δέντρα απόφασης για να αποφασίσει, οδηγεί σε υψηλότερη ακρίβεια.

Ο MLP Classifier, ακολουθεί, με αρκετά υψηλή ακρίβεια, κοντά στο RF Classifier, πράγμα που οφείλεται αρκετά και στο μέγεθος του dataset, από το οποίο εξαρτάται η απόδοση του.

Τελευταίος σε όλες τις προσεγγίσεις ήταν ο Gaussian Naïve Bayes. Ο GaussianNB, ήταν εξαιρετικά γρήγορος, επιτυγχάνοντας και μια ικανοποιητική ακρίβεια, ωστόσο λόγο του μεγάλου Dataset, αλλά και των αρκετών υποθέσεων που λαμβάνει ως προς την ανεξαρτησία των χαρακτηριστικών μεταξύ τους, δεν κατάφερε να πλησιάσει τα άλλα δυο μοντέλα. Έμεινε σταθερά σε ακρίβειες κάτω του 80%.

Ερώτημα 3:

Στο ερώτημα αυτό, προσπαθούμε στα πλαίσια της διεξαγωγής unsupervised learning, να ομαδοποιήσουμε τους χρήστες με βάση τις δραστηριότητες τους σε συστάδες.

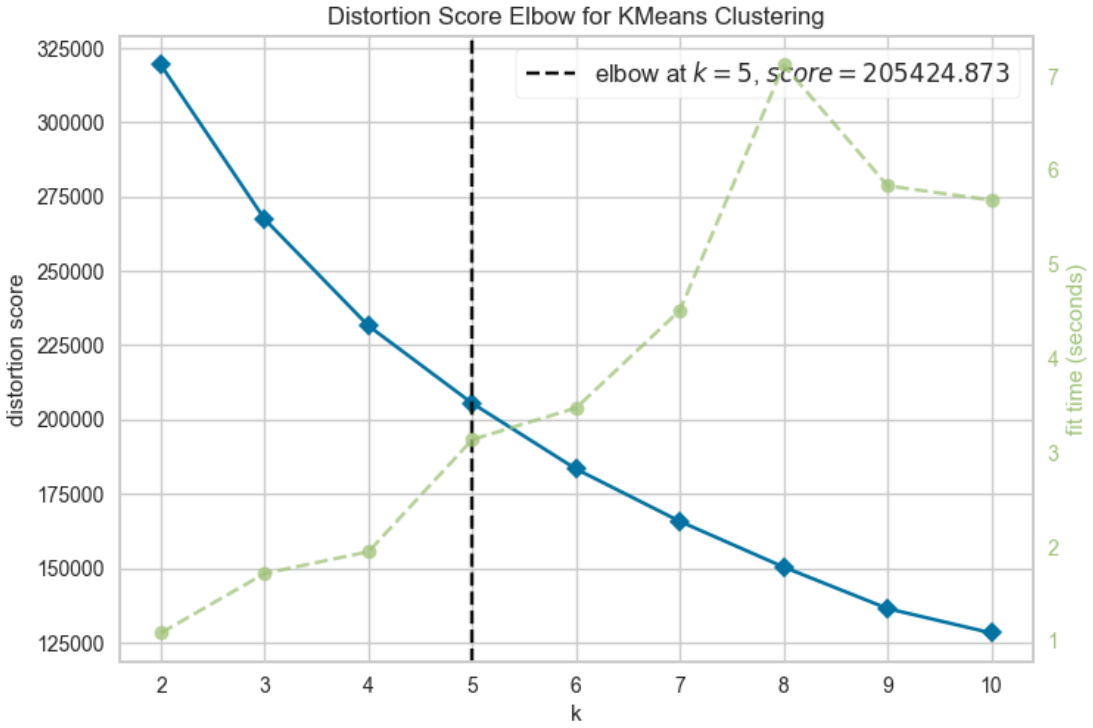
Για να το κάνουμε αυτό κάνουμε την απαραίτητη επεξεργασία του dataset (κανονικοποίηση, sampling, rolling window). Αρχικά κάνουμε drop το label από το data frame που θα αξιοποιήσουμε στους αλγορίθμους συσταδοποίησης. Ωστόσο, αποφάσισα να κρατήσω σε ένα διαφορετικό data frame, τα label αυτά για να προσπαθήσω αργότερα να ερμηνεύσω την συσταδοποίηση που πραγματοποιούν οι αλγόριθμοι που επιλέχθηκαν.

KMeans

Ο αλγόριθμος KMeans, ίσως ο πιο ευρέως διαδεδομένος αλγόριθμος clustering, επιλέχθηκε αρχικά.

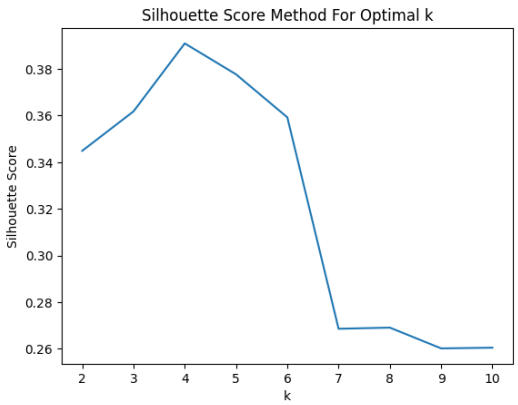
Ωστόσο για να εκτελέσουμε τον αλγόριθμο αυτό πρέπει να αποφασίσουμε την τιμή της παραμέτρου k (= ο αριθμός των cluster).

Για να το κάνουμε αυτό, επέλεξα δύο διαφορετικούς τρόπους προσέγγισης της παραμέτρου k. Πρώτα με την χρήση του KElbowVisualizer, όπως είχαμε δει και στο φροντιστήριο του μαθήματος, είδα ότι η παράμετρος που αυτή η μέθοδος μου επιδεικνύει είναι η k=5.



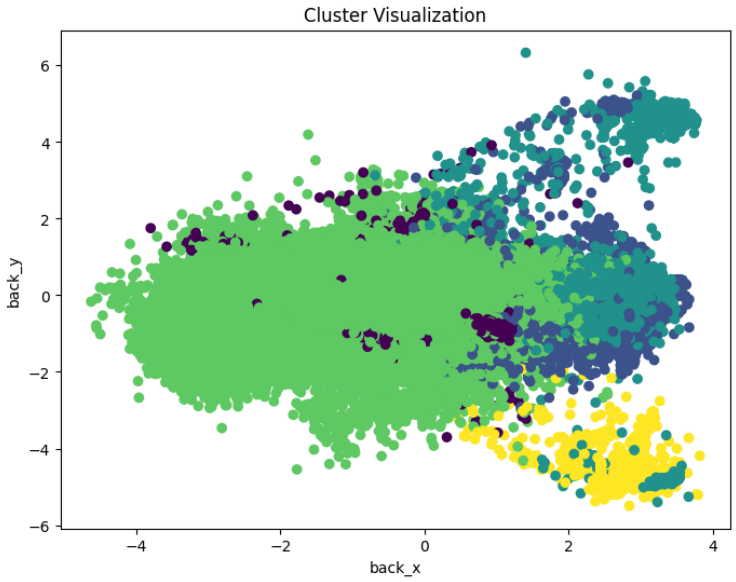
Ωστόσο επειδή αυτό το γράφημα δεν αναδεικνύει κάποιο ξεκάθαρο Elbow, προσπάθησα να αξιοποιήσω και άλλες μεθόδους προσδιορισμού, όπως πχ μέσω του Silhouette Score.

Εκτελώντας δοκιμές για k από 2 έως 11 και υπολογίζοντας την συγκεκριμένη μετρική, αυτή μου υπέδειξε k=4.



Έτσι προχώρησα, στην διεξαγωγή Clustering και με τις 2 τιμές και ερμήνευσα τα αποτελέσματα τους με βάση τις τιμές των label που είχα κρατήσει νωρίτερα και υπολόγισα και κάποιες μετρικές που φαίνονται στο ipynb.

KMeans για k=5 (value from KElbow)



cluster	0	1	2	3	4
review					
1	3873	493	137	115196	4
2	1277	3367	2259	22229	4
3	572	53	70	24817	0
4	466	3	32	7112	0
5	142	5	9	6560	1
6	1465	169	34	72686	0
7	222605	61643	4276	1688	58
8	75	11742	20328	18	10725
13	7869	1810	97	29620	0
14	133	261	166	5018	0
130	341	74	37	3727	0
140	50	15	0	722	0

Σχολιασμός Αποτελεσμάτων

Παρατηρούμε, ότι εδώ έχουμε όλες τις δραστηριότητες που περιλαμβάνουν κίνηση στο cluster 3.

1. walking
2. running
3. shuffling
4. stairs (ascending)
5. stairs (descending)
6. standing

Η κλάση που ο χρήστης είναι καθιστός (7), παρατηρούμε ότι έχει τοποθετηθεί στο cluster 0.

Η κλάση που ο χρήστης είναι ξαπλωμένος (8) στο cluster 2.

Τέλος, οι τελευταίες 4 κλάσεις:

13. cycling (sit)

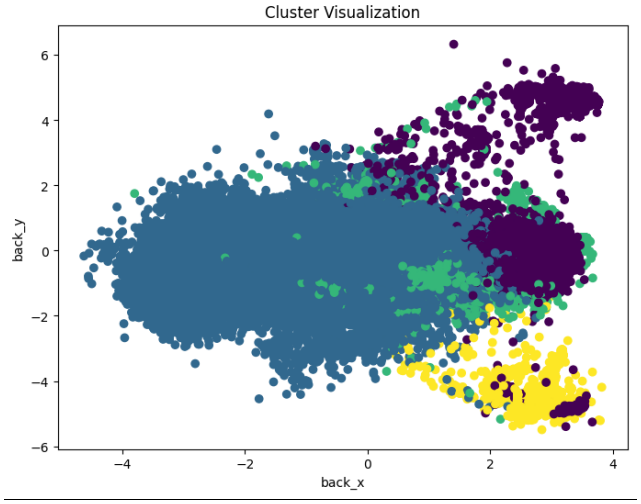
14. cycling (stand)

130. cycling (sit, inactive)
140. cycling (stand, inactive)

που ο χρήστης κάνει ποδήλατο τοποθετούνται κατά κύριο λόγο στην κλάση των κινητικών δραστηριοτήτων (cluster 3). Ωστόσο, βλέπουμε μια τάση στην κλάση 13 και 130 όπου ο χρήστης κάνει ποδήλατο καθιστός, να τοποθετεί αρκετά δείγματα και εκεί (cluster 0), αντί για το 3ο cluster.

Τα cluster 1 και 4, παρόλο που έχουν κάποια δείγματα, δεν μπορούμε να παρατηρήσουμε με βάση τα labels κάποιο μοτίβο.

KMeans για k=4 (value from Silhouette Score)



cluster	0	1	2	3
review				
1	3567	24	116108	4
2	4028	3	25096	9
3	442	11	25059	0
4	249	0	7364	0
5	72	0	6644	1
6	1040	5	73309	0
7	283897	4	6311	58
8	11761	10100	10292	10735
13	3488	9	35899	0
14	113	0	5465	0
130	120	0	4059	0
140	14	0	773	0

Σχολιασμός Αποτελεσμάτων:

Παρατηρούμε ότι εδώ έχουμε όλες τις δραστηριότητες που περιλαμβάνουν κίνηση στο cluster 2.

1: walking

2: running

3: shuffling

4: stairs (ascending)

5: stairs (descending)

6: standing

Η κλάση που ο χρήστης είναι καθιστός (7), παρατηρούμε ότι έχει τοποθετηθεί στο cluster 0, και αναγνωρίζεται σε μεγάλο ποσοστό.

Η κλάση που ο χρήστης είναι ξαπλωμένος (8), το μοντέλο δυσκολεύεται να την ξεχωρίσει σε αντίθεση με το παραπάνω παράδειγμα με 5 clusters, και κατανέμει περίπου ισόποσα τα στοιχεία του σε όλα τα clusters.

Τέλος, οι τελευταίες 4 κλάσεις:

13: cycling (sit)

14: cycling (stand)

130: cycling (sit, inactive)

140: cycling (stand, inactive)

Που ο χρήστης κάνει ποδήλατο τοποθετούνται κατά κύριο λόγο στην κλάση των κινητικών δραστηριοτήτων (cluster 2). Ωστόσο, βλέπουμε μια τάση στην κλάση 13 και 130 όπου ο χρήστης κάνει ποδήλατο καθιστός, να τοποθετεί αρκετά δείγματα και εκεί (cluster 0), αντί για το cluster 2.

Τα cluster 1 και 3, παρόλο που έχουν κάποια δείγματα, δεν μπορούμε να παρατηρήσουμε με βάση τα labels κάποιο μοτίβο.

Agglomerative Clustering:

Ο δεύτερος αλγόριθμος clustering που αξιοποίησα ήταν ο Agglomerative Clustering, τις παραμέτρους του οποίου προσπάθησα να τις υπολογίσω μέσω δοκιμών και να παρώ αυτή με το μεγαλύτερο Silhouette Score.

Έλαβα n_clusters=4.

	Parameter	Silhouette Score
KMeans	k=4	0.4530
	k=5	0.4631
Agglomerative Clustering	n_clusters=2	0.3052
	n_clusters=3	0.3229
	n_clusters=4	0.3492
	n_clusters=5	0.3162

Συγκρίνοντας τους δύο αλγορίθμους παρατηρώ ότι ο KMeans, είναι καλύτερος στην ομαδοποίηση των δεδομένων, πράγμα που πιθανών οφείλεται στο scalability του σε μεγαλύτερα σύνολα δεδομένων. Επιπλέον είναι αρκετά πιο γρήγορος, από τον ιεραρχικό αλγόριθμο Agglomerative Clustering.

Το τελευταίο ήταν και ο λόγος που στον Agglomerative Clustering, χρειάστηκε ακόμα πιο έντονη δειγματοληψία, πράγμα που δυσκολεύει και την εξαγωγή ασφαλών συμπερασμάτων.

Δυστυχώς, στο κομμάτι αυτό της εργασίας (ερώτημα 3), δεν κατάφερα να επιτύχω καλά αποτελέσματα διότι όποτε πείραζα παραμέτρους που θα οδηγούσαν σε καλύτερα αποτελέσματα, το σύστημα μου δεν μπορούσε να ανταποκριθεί στην υπολογιστική πολυπλοκότητα. Επιπλέον, συνάντησα μεγάλη δυσκολία, στην προσπάθεια μου να δοκιμάσω και τον DBScan αλγόριθμο, λαμβάνοντας διαρκώς σφάλματα μνήμης κ.α, ακόμα και σε μικρά σύνολα δεδομένων.