Αλέξανδρος Ξιάρχος
st1059619@ceid.upatras.gr
1059619

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ ΜΑΘΗΣΗΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ · 2023–2024

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΡΩΤΗΜΑ 1

Το σύνολο δεδομένων περιλαμβάνει 22 `.csv` αρχεία που αντιστοιχούν σε 22 συμμετέχοντες. Σύμφωνα με την περιγραφή του dataset, περιλαμβάνεται η στήλη `timestamp`, με την ημερομηνία και ώρα, οι στήλες $\texttt{back}_{x,y,z}$ και $\texttt{thigh}_{x,y,z}$ με τις τιμές του κάθε αισθητήρα για κάθε διάσταση, και η στήλη `label`, η οποία προσδιορίζει τη δραστηριότητα του συμμετέχοντα τη δεδομένη στιγμή.

Η στήλη `label` παίρνει τις εξής τιμές:

```
1 - Walking
2 - Running
3 - Shuffling
4 - Stairs (ascending)
5 - Stairs (descending)
6 - Standing
8: lying
13 - Cycling (sit)
14 - Cycling (stand)
130 - Cycling (sit, inactive)
140 - Cycling (stand, inactive)
```

Για την εισαγωγή και την προεπεξεργασία των αρχείων, θα χρησιμοποιήσουμε τη βιβλιοθήκη `pandas` ενώ για την οπτικοποίηση τις βιβλιοθήκες `matplotlib` και `seaborn` της Python.

1.1 ΑΝΑΛΥΣΗ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ

Εισάγουμε τα `.csv` αρχεία μέσω της `os` βιβλιοθήκης και της `read_csv()` συνάρτησης. Καταρχάς, χρησιμοποιώντας τη `head()` μπορούμε να δούμε τις πρώτες εγγραφές του dataset μας. Για παράδειγμα για το πρώτο αρχείο του συνόλου δεδομένων `S006.csv`:

	timestamp	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
0	2019-01-12 00:00:00.000	-0.760242	0.299570	0.468570	-5.092732	-0.298644	0.709439	6
1	2019-01-12 00:00:00.010	-0.530138	0.281880	0.319987	0.900547	0.286944	0.340309	6
2	2019-01-12 00:00:00.020	-1.170922	0.186353	-0.167010	-0.035442	-0.078423	-0.515212	6
3	2019-01-12 00:00:00.030	-0.648772	0.016579	-0.054284	-1.554248	-0.950978	-0.221140	6
4	2019-01-12 00:00:00.040	-0.355071	-0.051831	-0.113419	-0.547471	0.140903	-0.653782	6

Μέσω της `info()` εξάγουμε το συμπέρασμα πώς για κάθε χρονική στιγμή δίνονται οι τιμές των αισθητήρων, αποθηκευμένες ως `float24`, στις τρεις διαστάσεις (x, y, z) για τις περιοχές της πλάτης και του μηρού, καθώς και ένα `int64` για το `label`. Παρατηρείται η ίδια μορφολογία σε όλα τα `.csv` του συνόλου δεδομένων, με κάποιες διαφοροποιήσεις που θα αναλυθούν στη συνέχεια.

Παρόλο που στην περιγραφή αναφέρεται πως δεν υπάρχουν missing values, για να ελέγξουμε την ακεραιότητα του dataset, μέσω της συνάρτησης `concat()` ενώνουμε όλα τα 22 αρχεία σε ένα ενιαίο dataframe, `df_combined`. Τρέχοντας την `isnull().sum()`, έχουμε:

	sum
timestamp	0
$\texttt{back}_x$	0
$\texttt{back}_y$	0
$\texttt{back}_z$	0
$\texttt{thigh}_x$	0
$\texttt{thigh}_y$	0
$\texttt{thigh}_z$	0
label	0
index	5740689
Unnamed: 0	6323682

Παρατηρούμε πως στις στήλες $\texttt{back}_{x,y,z}$ και $\texttt{thigh}_{x,y,z}$, οι οποίες είναι και αυτές που μας ενδιαφέρουν, όντως δεν παρατηρούνται missing values. Όμως, έχουν εμφανιστεί `NaN` τιμές στις στήλες `"index"` και `"Unnamed: 0"`, οι οποίες στήλες μάλλον θα εμφανίζονται επιπλέον σε κάποια αρχεία.

Ελέγχοντας όλα τα αρχεία, η στήλη `"index"` εμφανίζεται στα αρχεία `S015.csv` και `S021.csv` και η στήλη `"Unnamed: 0"` στο αρχείο `S023.csv`. Έπειτα από έλεγχο, φαίνεται πως πρόκειται για δείκτες αύξουσας αρίθμησης που δεν προσφέρουν κάποια χρήσιμη πληροφορία. Επομένως, μπορούμε να τις αφαιρέσουμε χρησιμοποιώντας τη συνάρτηση `drop('όνομα', axis=1)`. Τα επεξεργασμένα `.csv` αρχεία αποθηκεύονται με το επίθεμα `fix`.

Χρησιμοποιώντας τη συνάρτηση `describe()` μπορούμε να υπολογίσουμε βασικές στατιστικές μετρικές για τα δεδομένα μας. Η συνάρτηση επιστρέφει ένα dataframe με τις ακόλουθες στήλες:

- **`count`**: ο συνολικός αριθμός των μη-μηδενικών τιμών για κάθε στήλη.
- **`mean`**: ο μέσος όρος των τιμών για κάθε στήλη.
- **`min`**: η ελάχιστη τιμή για κάθε στήλη.
- **`25%`**: η τιμή του 25ου εκατοστημορίου για κάθε στήλη.
- **`50%`**: η τιμή του 50ου εκατοστημορίου για κάθε στήλη.
- **`75%`**: η τιμή του 75ου εκατοστημορίου για κάθε στήλη.
- **`max`**: η μέγιστη τιμή για κάθε στήλη.

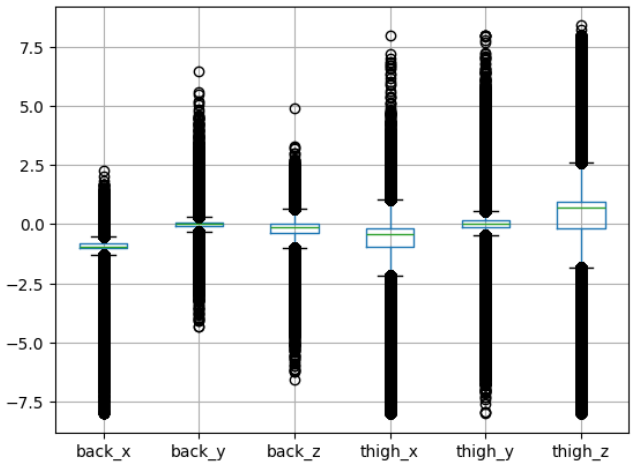
Ενώνοντας συγκεντρωτικά τις μετρήσεις όλων των συμμετεχόντων στο `df_combined`, αυτά είναι τα **βασικά συγκεντρωτικά στατιστικά μεγέθη** όπως προκύπτουν από το `describe()` για όλες τις μετρήσεις από τους συμμετέχοντες, έχοντας αφαιρέσει την ετικέτα `label` μιας και αποτελείται από κατηγορικές τιμές:

	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z
count	6461328	6461328	6461328	6461328	6461328	6461328
mean	-0.884957	-0.013261	-0.169378	-0.594888	0.020877	0.374916
std	0.377592	0.231171	0.364738	0.626347	0.388451	0.736098
min	-8.000000	-4.307617	-6.574463	-8.000000	-7.997314	-8.000000
25%	-1.002393	-0.083129	-0.372070	-0.974211	-0.100087	-0.155714
50%	-0.974900	0.002594	-0.137451	-0.421731	0.032629	0.700439
75%	-0.812303	0.072510	0.046473	-0.167876	0.154951	0.948675
max	2.291708	6.491943	4.909483	7.999756	7.999756	8.406235

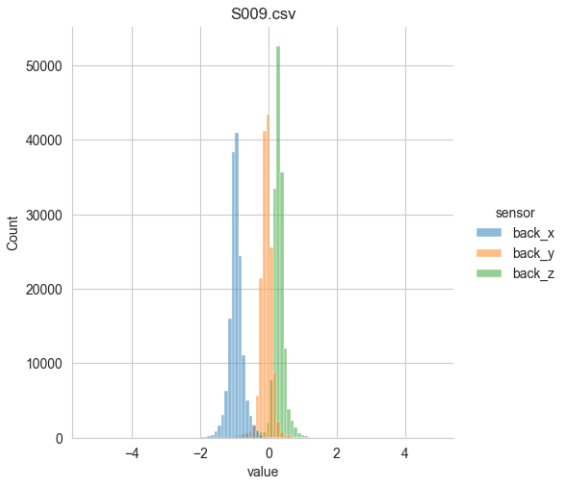
Ως αρχικές παρατηρήσεις, βλέπουμε πως οι τιμές βρίσκονται στο διάστημα $[-8, 8]$, ενώ η τυπική τους απόκλιση είναι μικρή, το οποίο δείχνει ότι οι μετρήσεις είναι αρκετά συμπυκνωμένες γύρω από τον μέσο όρο που είναι κοντά στο μηδέν. Προφανώς ελέγχοντας κάθε συμμετέχοντα ξεχωριστά, μπορεί να διεξαχθούν αντίστοιχα συμπεράσματα.

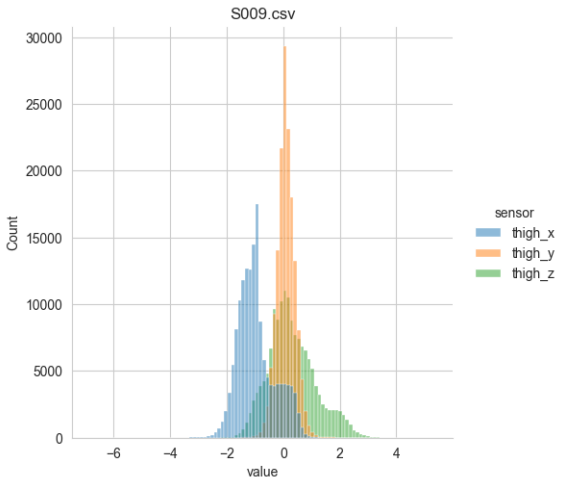
1.2 ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ

Μέσω της `plotbox()` της `Matplotlib`, μπορούμε να δημιουργήσουμε το διάγραμμα των τιμών της `df_combined` για μια πρώτη οπτικοποίηση των δεδομένων:

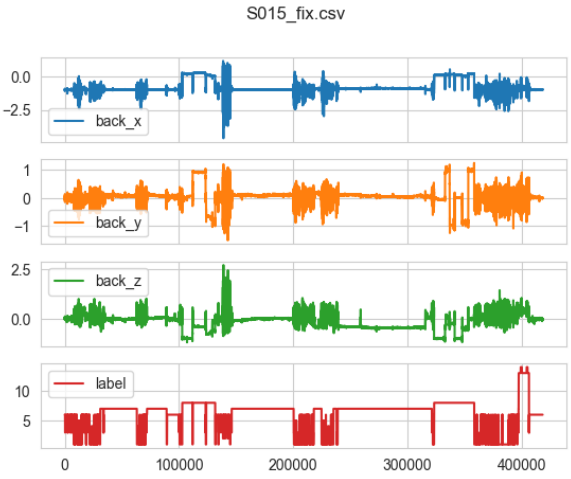


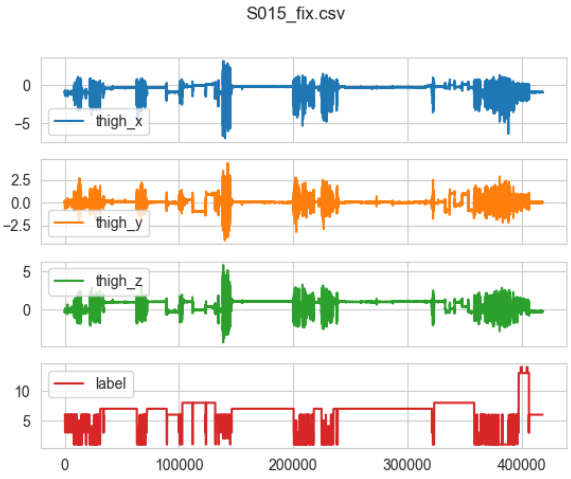
Πέρα από τις προηγούμενες παρατηρήσεις που επιβεβαιώνονται, επιπλέον παρατηρούμε μια συμμετρικότητα κοντά στο μηδέν για κάθε διάσταση. Επίσης, χρησιμοποιώντας την `displot()`, μπορούμε να οπτικοποιήσουμε την κατανομή των τιμών. Ενδεικτικά για τον `S009`:





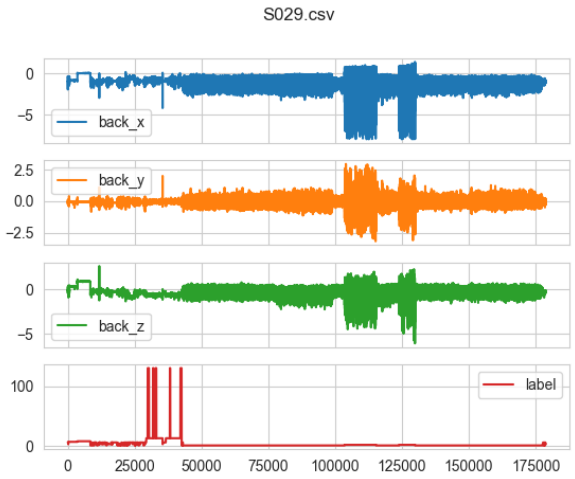
Χρησιμοποιώντας την `plot()`, μπορούμε να δημιουργήσουμε subplots για τις στήλες $\texttt{back}_{x,y,z}$ και $\texttt{thigh}_{x,y,z}$. Αυτά, για παράδειγμα, είναι τα subplots για τον συμμετέχοντα `S015`:

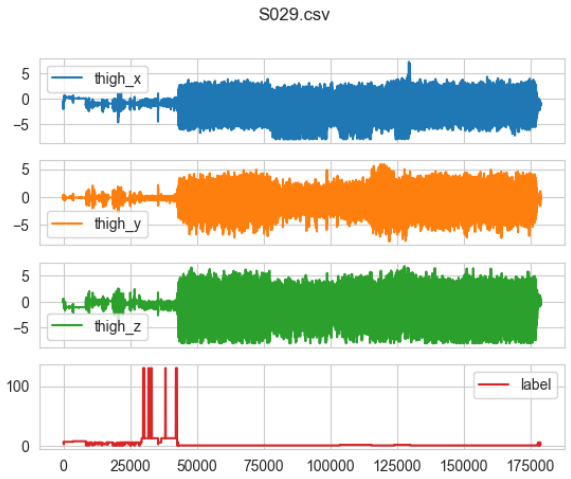




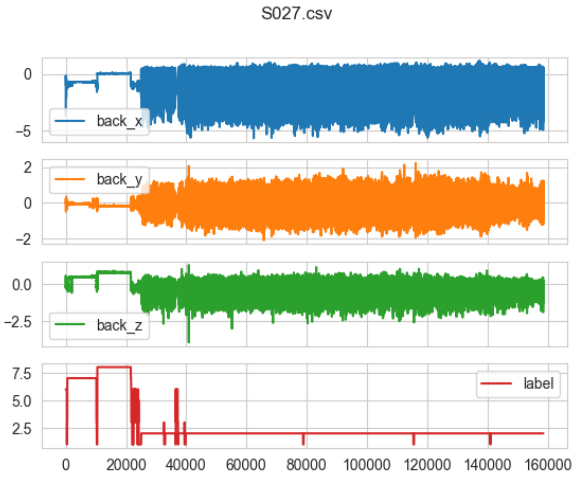
Φαίνεται ότι ο συμμετέχοντας κατά τη διάρκεια της μέτρησης μεταβάλλει τη φυσική του δραστηριότητα (μάλιστα με παρόμοιο τρόπο σε πλάτη και μηρό), καθώς υπάρχουν στιγμές που δεν υπάρχουν έντονες διακυμάνσεις των τιμών των μετρήσεων των αισθητήρων και άλλες όπου είναι πιο ενεργός, με την τιμή του `label` να αλλάζει και αυτή. Στις στιγμές που ο συμμετέχοντας δεν κινείται, το `label` φαίνεται να παίρνει την τιμή `8 - Standing`, που επιβεβαιώνει τη στασιμότητά του.

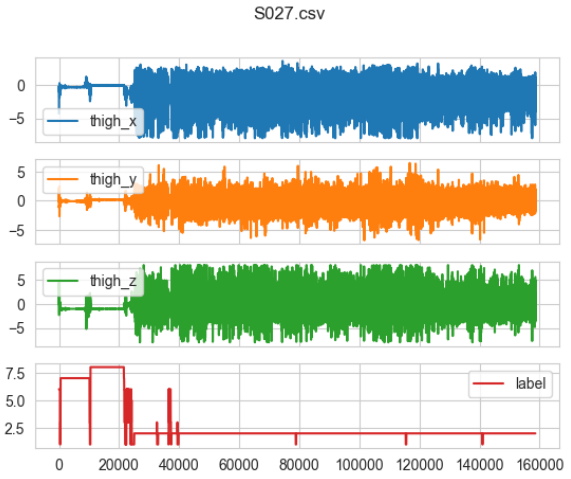
Από την άλλη, στον συμμετέχοντα `S029` παρατηρούμε πως η κίνηση της πλάτης δεν ταυτίζεται με την (έντονη) κίνηση των μηρών, κάτι που μας οδηγεί στο συμπέρασμα πως ο συμμετέχοντας κάνει ποδήλατο. Το γεγονός αυτό επιβεβαιώνεται και από τα spikes του `label` στις τιμές 100.



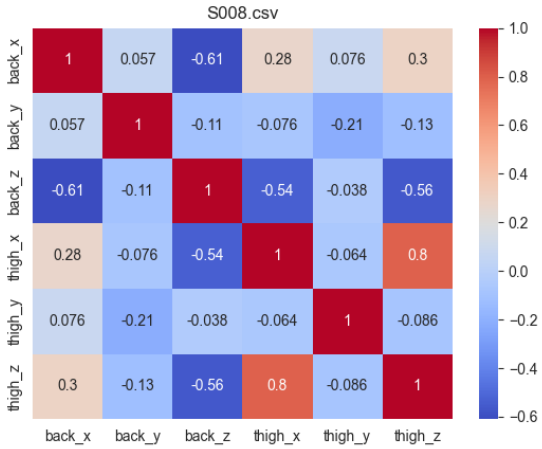


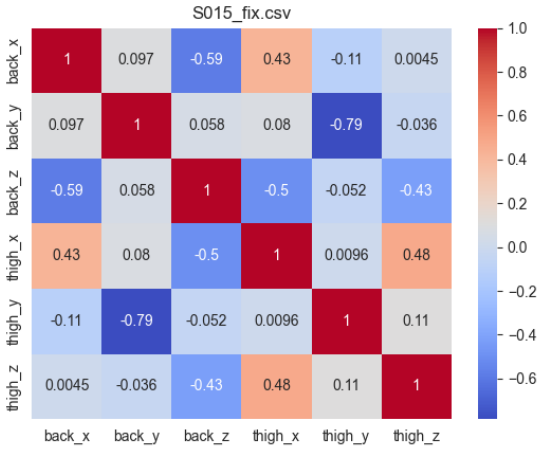
Τέλος, για τον συμμετέχοντα `S027`, φαίνεται να έχει μια πολύ έντονη φυσική δραστηριότητα με τη `label` να παραμένει σταθερή με τιμή κοντά στο 2.5, κάτι από το οποίο μπορούμε να συμπεράνουμε πως ο συμμετέχοντας τρέχει:



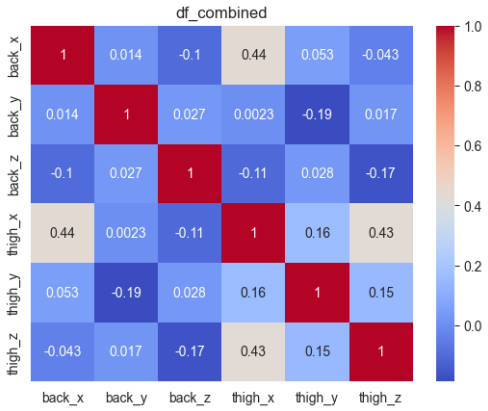


Τέλος, για τον εντοπισμό συσχετίσεων, μπορούμε να δημιουργήσουμε ένα `heatmap()` μέσω της `seaborn`. Για παράδειγμα, για τον συμμετέχοντα `S008` φαίνεται πως υπάρχει μια κάποια συσχέτιση ανάμεσα στις στήλες `back_x` + `back_z`, `thigh_x` + `thigh_z` και `back_z` + `thigh_x` ενώ στον `S015` ανάμεσα στις στήλες `back_x` + `back_z` και `back_y` + `thigh_y`.





Δημιουργώντας heatmap και για το συγκεντρωτικό dataframe `df_combined`, παρατηρούμε μια αμυδρή συσχέτιση ανάμεσα στα `thigh_x` + `back_x` και `thigh_x` + `thigh_z`.



> Στο παράρτημα παρατίθενται τα plots και τα heatmaps για όλους τους συμμετέχοντες.

2 ΕΡΩΤΗΜΑ 2

2.1 ΟΡΙΣΜΟΣ ΤΑΞΙΝΟΜΗΤΩΝ

Στην συνάρτηση `get_classifier(option)` ορίζονται και μπορούν να επιλεχθούν οι ταξινομητές που θα χρησιμοποιηθούν στη συνέχεια.

Σε κάθε περίπτωση ταξινομητή, σε καθένα από τα 22 dataframes του dataset αφαιρείται η στήλη `timestamp`, και το dataframe διαχωρίζεται από τη στήλη `label` στα `X` και `Y`. Όταν γίνει ο διαχωρισμός, γίνονται split στα dataframes `X_train`, `X_test`, `Y_train`, `Y_test` με `test_size = 0.3`.

Ο `MLPClassifier` τρέχει με `max_iter=500`, και ο `RandomForestClassifier` με `n_estimators=100, criterion='entropy'`.

Αφού γίνει η επιλογή του classifier, αυτός γίνεται train μέσω της `fit(X_train, Y_train)` και αποθηκεύονται τα predictions του μέσω της `predict(X_test)`.

Το αποτέλεσμα είναι να εκτελούνται και οι τρεις classifiers για όλα τα αρχεία και να αποθηκεύονται το `training accuracy` όπως επίσης οι μετρικές μέσω της `sklearn.metrics`: `testing accuracy`, `precision`, `recall` και `f1`.

2.2 ΑΠΟΤΕΛΕΣΜΑΤΑ

Τρέχουμε κάθε classifier για όλους τους συμμετέχοντες:

2.2.1 NEURAL NETWORKS

file	training accuracy	testing accuracy
S006.csv	0.9141337173536156	0.9122442155399509
S008.csv	0.9329507794280103	0.9315337677112421
S009.csv	0.8956013466020495	0.8916271040138111
S010.csv	0.8519300925436921	0.8499360159249254
S012.csv	0.9738502515979364	0.9730834604488996
S013.csv	0.9007985198546176	0.8996423539612008
S014.csv	0.8991970063148047	0.8973778274987039
S015_fix.csv	0.9135976563300259	0.9136856865150815
S016.csv	0.9141337173536156	0.9447883255491156
S017.csv	0.9329507794280103	0.9109135048143804
S018.csv	0.8956013466020495	0.8708032518979748
S019.csv	0.8519300925436921	0.9578000537008861
S020.csv	0.9542851869085204	0.9545083401376414
S021_fix.csv	0.9347834306997145	0.9334105321202095
S022.csv	0.9020188641720371	0.9002873194379992
S023_fix.csv	0.9318747924277648	0.9256308422531119
S024.csv	0.9218583766848449	0.9191376243623073
S025.csv	0.8687010665187103	0.8650728577798875
S026.csv	0.8737227345922998	0.8699446645716628
S027.csv	0.9876495387719804	0.9867580292584497
S028.csv	0.9772106137134159	0.9761270533155749
S029.csv	0.9475463825229214	0.9443625850974541

2.2.2 RANDOM FOREST

file	training accuracy	testing accuracy
S006.csv	1.0	0.9306598810892809
S008.csv	1.0	0.9432683357598033
S009.csv	0.9999907513595502	0.8979499352611136
S010.csv	0.9999918750050781	0.8659367742547041
S012.csv	0.9999962643216569	0.9788363477881892
S013.csv	1.0	0.9186264947075612
S014.csv	1.0	0.9186264947075612
S015_fix.csv	0.9999863422495681	0.9243216112430089
S016.csv	1.0	0.9555080374392737
S017.csv	0.9999922065574026	0.9255794077266487
S018.csv	0.9999955671597462	0.8967336215634761
S019.csv	0.9999952052397141	0.9646916674125123
S020.csv	1.0	0.959730459672137
S021_fix.csv	1.0	0.9460601047697822
S022.csv	1.0	0.9111284446243619
S023_fix.csv	0.9999896213882431	0.9385867196202838
S024.csv	0.999983245792599	0.9339926897441411
S025.csv	0.999987670303927	0.8797307210978293
S026.csv	0.999992680427463	0.8916006285011614
S027.csv	1.0	0.9887968723726248
S028.csv	0.9999827025531032	0.9785890140049239
S029.csv	1.0	0.9555348316702416

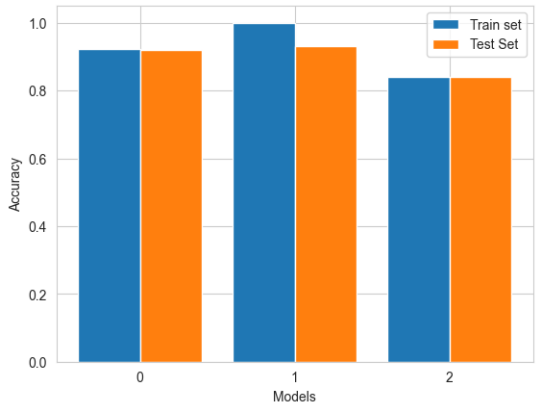
2.2.3 BAYESIAN NETWORKS

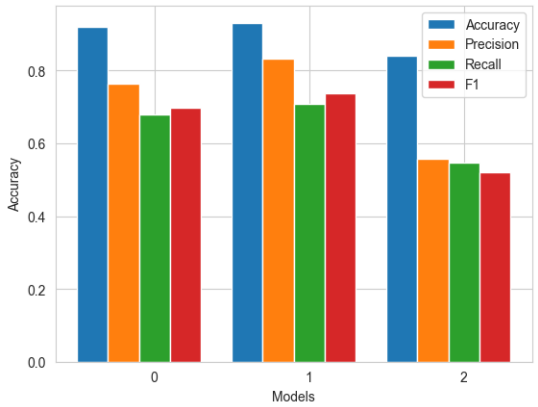
file	training accuracy	testing accuracy
S006.csv	0.8708720149879761	0.8709109148295858
S008.csv	0.8950056598884388	0.8940945289068156
S009.csv	0.8275868447338242	0.8281398359948209
S010.csv	0.7653420216612364	0.7660552632826201
S012.csv	0.9394446540575070	0.9392373066027457
S013.csv	0.8135999969034615	0.8122538925616849
S014.csv	0.8498128946752943	0.8503005993797011
S015_fix.csv	0.8722966190238806	0.8719307191000494
S016.csv	0.8932883694009454	0.8942753174647835
S017.csv	0.8588490643972162	0.8601511142631134
S018.csv	0.7839300675121571	0.7838997952048985
S019.csv	0.8906698759595514	0.8916248993108387
S020.csv	0.9124273688986991	0.9138529731087762
S021_fix.csv	0.8831981547652809	0.8823821339950372
S022.csv	0.8233673689600163	0.8252288188307777
S023_fix.csv	0.7625996346728662	0.7668668571705333
S024.csv	0.7699228468749215	0.7724438537166982
S025.csv	0.6912890697244313	0.6867043542053252
S026.csv	0.7031474161908945	0.7008300314250581
S027.csv	0.9621288555779763	0.9618715318648058
S028.csv	0.9570936829723933	0.9572385680267991
S029.csv	0.7504656237759890	0.7464329012403246

2.2.4 ΜΕΣΟΙ ΟΡΟΙ

Μέσω της `np.mean()` υπολογίζονται οι μέσοι όροι των αποτελεσμάτων των μοντέλων:

	classifier	score	accuracy	precision	recall	f1
1	MLPClassifier	0.9215787974	0.9197436955	0.7644127233	0.6806597787	0.6976677241
2	RandomForestClassifier	0.9999962322	0.9317139313	0.8322693957	0.7073846769	0.7384389196
3	GaussianNB	0.8403413636	0.8403609059	0.5568391564	0.5460153011	0.5208372416





2.3 ΣΥΜΠΕΡΑΣΜΑΤΑ

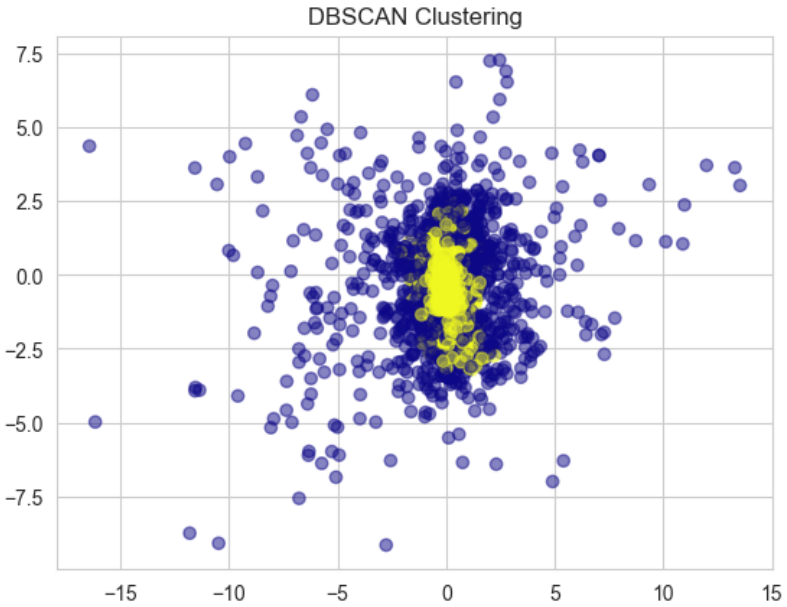
Παρατηρούμε πως το training accuracy του Random Forest είναι σχεδόν τέλειο, και μεγαλύτερο από το testing accuracy. Αυτό μας οδηγεί στο συμπέρασμα πως είναι πιθανό να υπάρχει overfitting· υπάρχει υπερβολικά καλή απόδοση στα training δεδομένα και δεν γενικεύει αρκετά ώστε να ανταποκρίνεται στα test δεδομένα. Σε συνδυασμό με την μικρότερη ακρίβεια των Bayesian Networks, η καλύτερη δυνατή επιλογή είναι τα Neural Networks.

3 ΕΡΩΤΗΜΑ 3

Για αρχή, θα χρησιμοποιήσουμε τον KMeans αλγόριθμο της `sklearn.cluster` για τη συσταδοποίηση. Καταρχάς περνάμε τα δεδομένα μέσα από έναν scaler, συγκεκριμένα τον `StandardScaler()`, ο οποίος τα κανονικοποιεί για να είναι ομοιόμορφα. Η διαδικασία αυτή είναι απαραίτητη, μιας και ο αλγόριθμος είναι επιρρεπής στις μικροαλλαγές των δεδομένων, και χρειάζεται η συστηματοποίησή τους.

Μιας και έχουμε πολυδιάστατα δεδομένα, είναι απαραίτητη η μείωση των διαστάσεων, ώστε να είναι εφικτό να απεικονιστούν στο δισδιάστατο επίπεδο. Για αυτό το λόγο χρησιμοποιούμε τον PCA της `sklearn.decomposition`. Εν τέλει, ο KMeans αλγόριθμος δημιουργεί τις επόμενες 5 συστάδες:

Ένας εναλλακτικός τρόπος για τη συσταδοποίηση είναι να χρησιμοποιήσουμε τον DBSCAN αλγόριθμο πάλι της `sklearn.cluster`. Ο DBSCAN δημιουργεί συστάδες οποιουδήποτε σχήματος, ενώ δεν επηρεάζεται τόσο από τα outliers. Δεν προκαθορίζεται ο αριθμός των συστάδων· καθορίζεται ένας `eps` αριθμός που ρυθμίζει τη μέγιστη απόσταση δύο δειγμάτων που ανήκουν στην ίδια συστάδα. Είναι υπολογιστικά πιο αργός, αλλά λέγεται πως επιφέρει καλύτερα αποτελέσματα.



> Λόγω του πολύ μεγάλου συνόλου δεδομένων και του μεγάλου χρόνου υπολογισμού του, ο αλγόριθμος εκτελέστηκε μόνο σε ένα `.csv` αρχείο αντί για το `df_combined`.

3.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

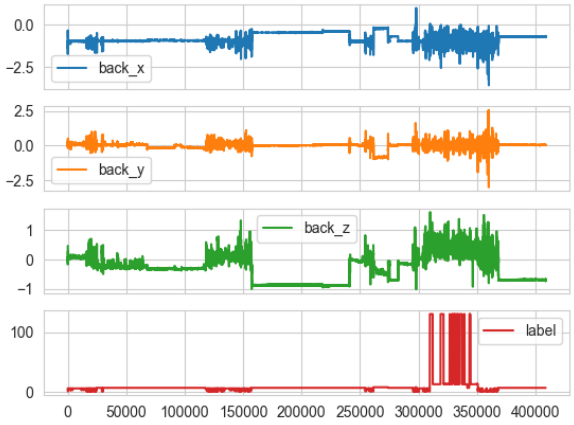
Αν και ο αλγόριθμος παραμετροποιήθηκε αρκετά και δοκιμάστηκαν διαφορετικές τιμές του `eps`, πέρα από την πολύ αρχή εκτέλεσή του, δε δημιούργησε ξεκάθαρες πλειάδες, παρόλο που εκτελέστηκε σε ένα πολύ μικρότερο σύνολο σε σύγκριση με τον KMeans. Σε αυτό ίσως ευθύνονται οι πολλαπλές διαστάσεις των δεδομένων (curse of dimensionality). Αντίθετα ο KMeans δημιούργησε άμεσα ξεκάθαρες πλειάδες σε πολύ σύντομο χρονικό διάστημα σε ένα μεγάλο πλήθος δεδομένων.

Εν τέλει, λόγω της μορφολογίας του συγκεκριμένου συνόλου δεδομένων, που δεν περιλαμβάνει θόρυβο ή πολλά outliers, ο KMeans ανταποκρίνεται καλύτερα.

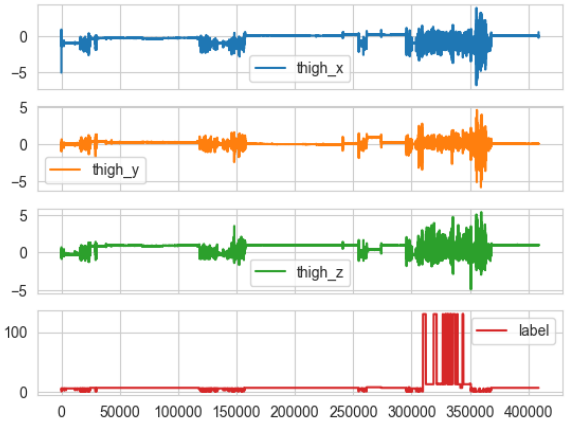
4 ΠΑΡΑΡΤΗΜΑ

4.1 PLOTS

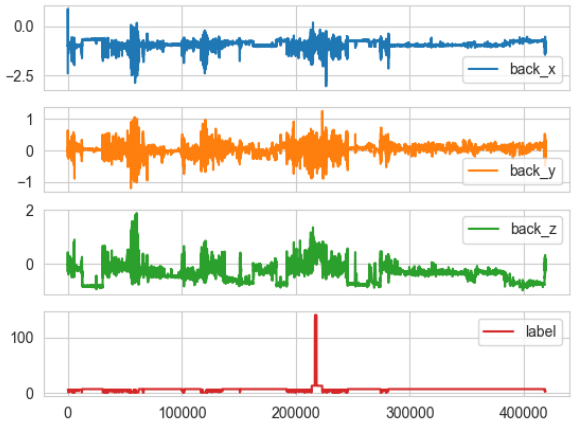
S006.csv



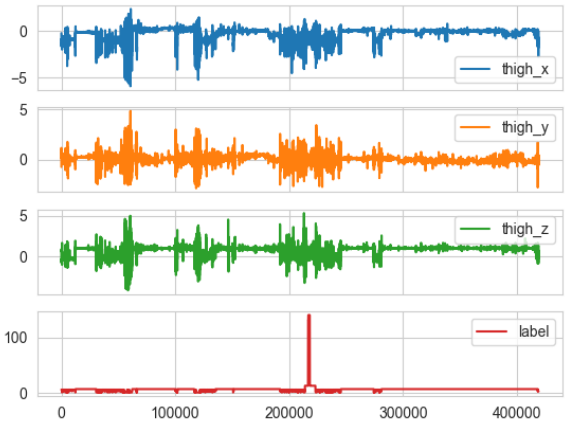
S006.csv



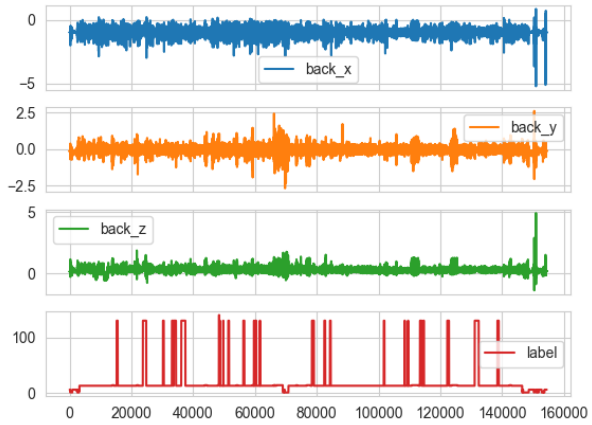
S008.csv



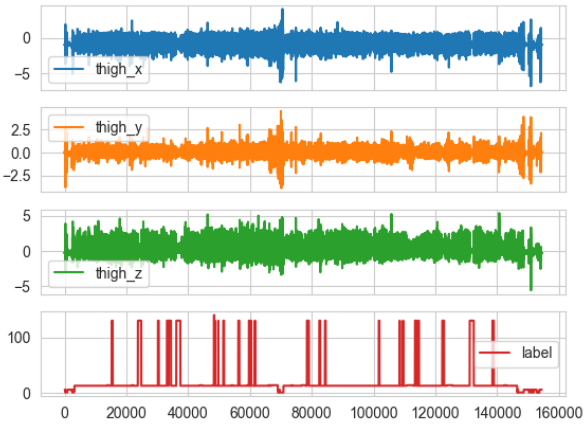
S008.csv



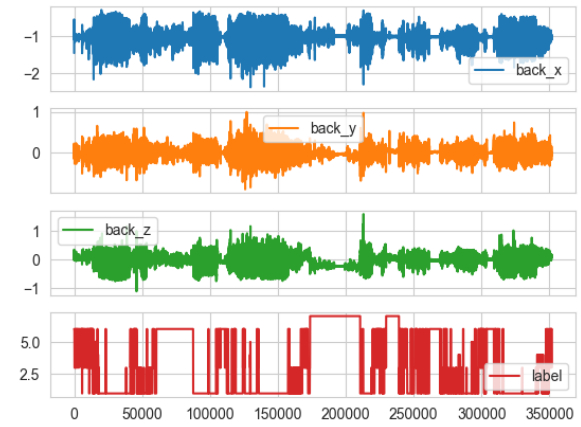
S009.csv



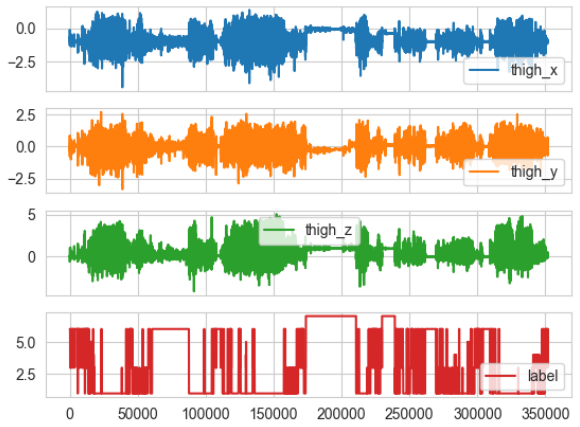
S009.csv



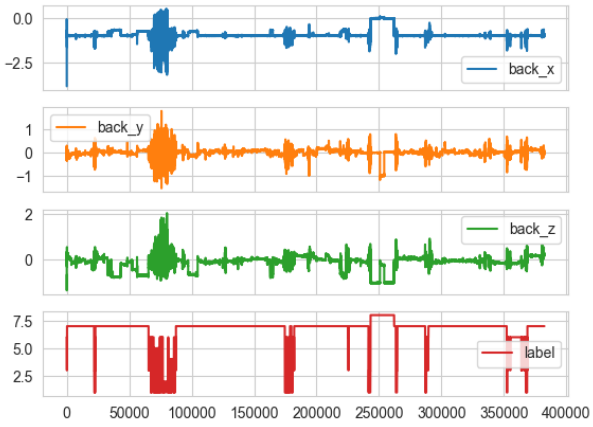
S010.csv



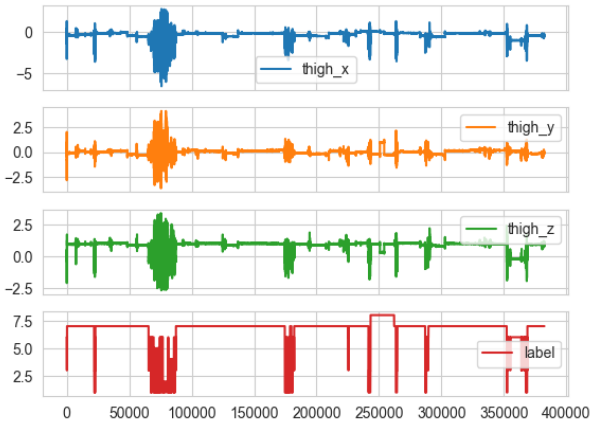
S010.csv



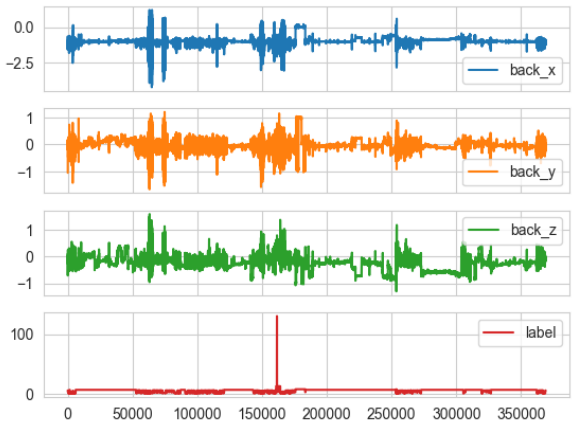
S012.csv



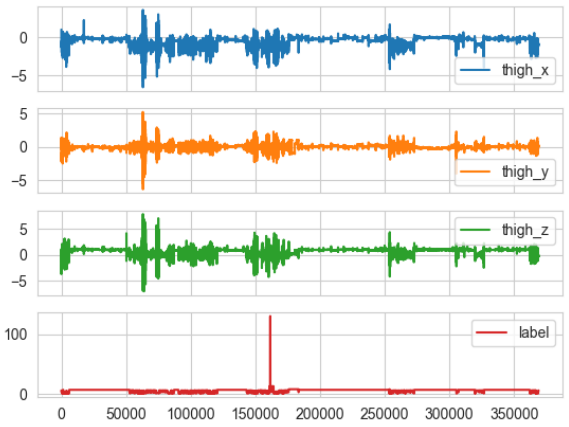
S012.csv



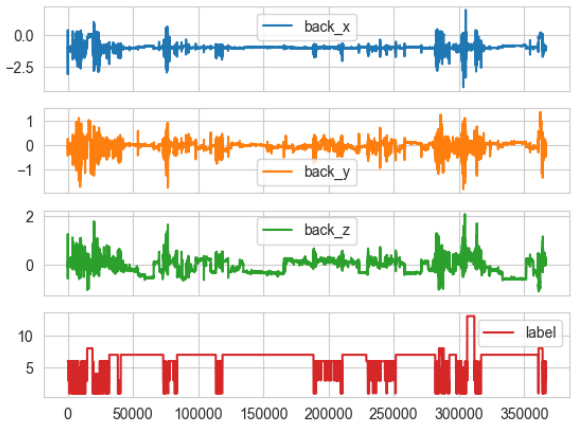
S013.csv



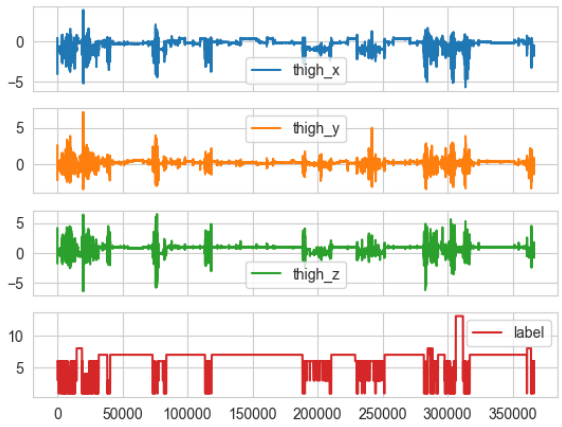
S013.csv



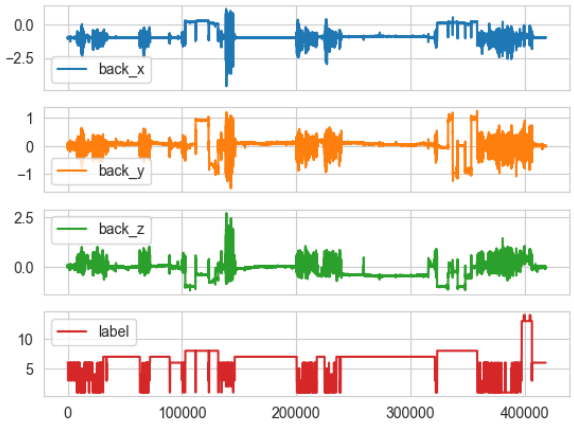
S014.csv



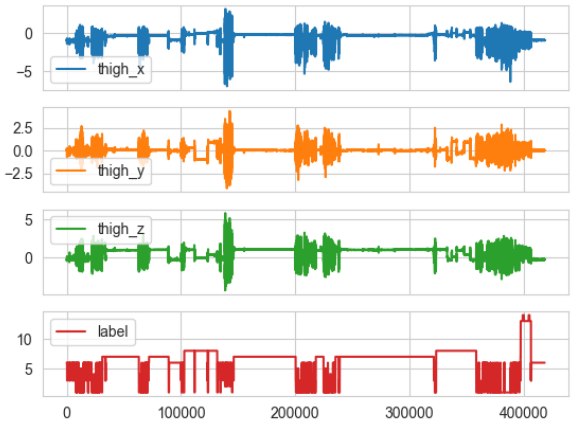
S014.csv



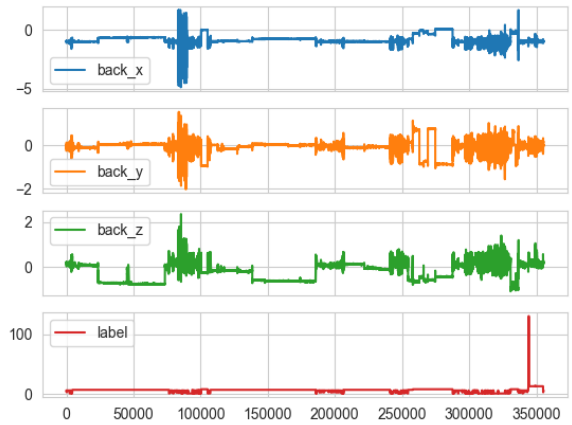
S015_fix.csv



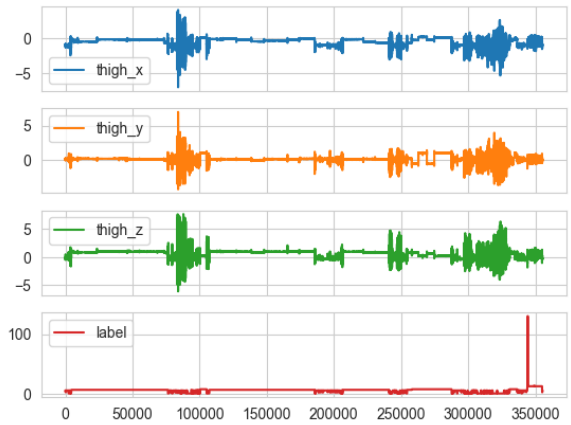
S015_fix.csv

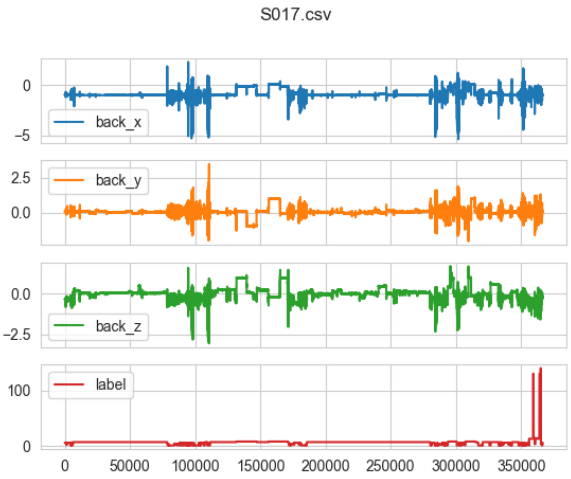


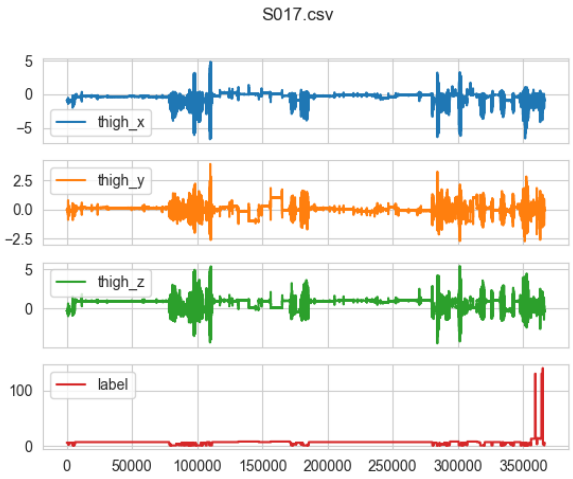
S016.csv

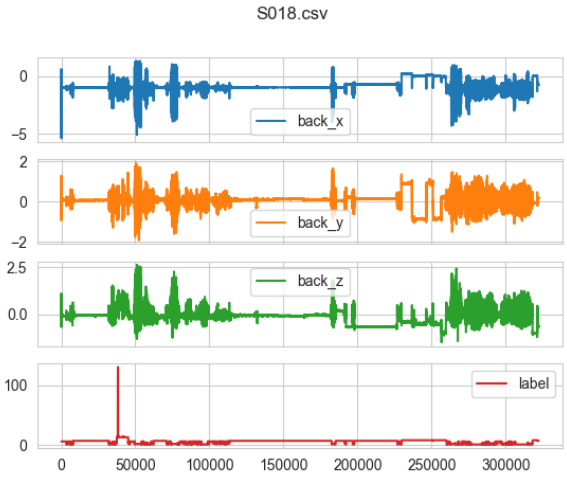


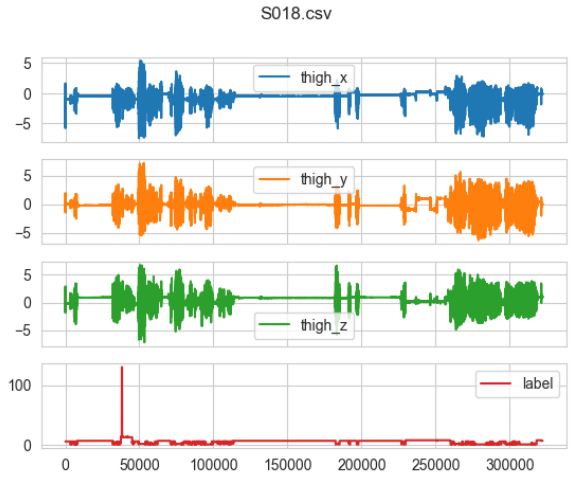
S016.csv

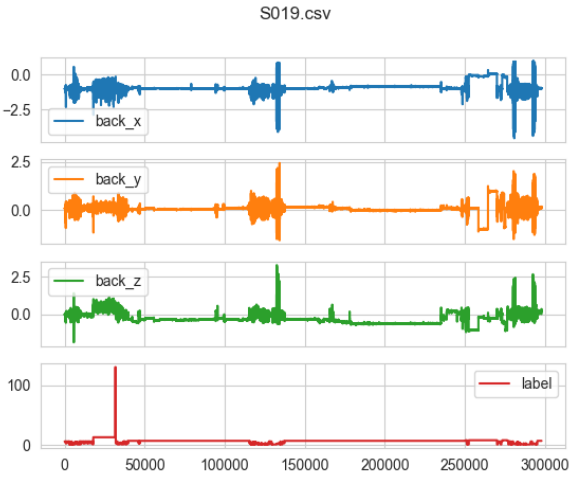


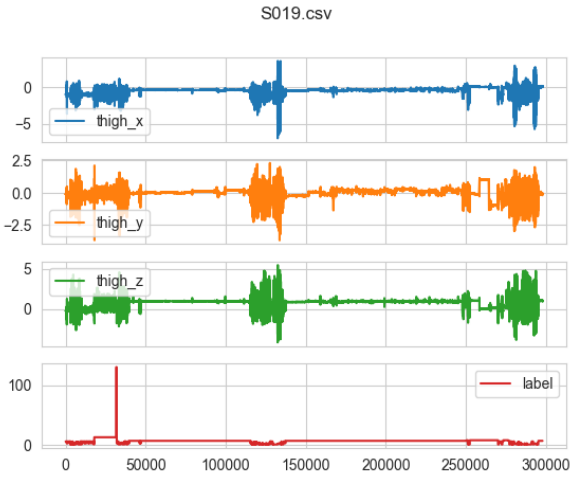

S017.csv
back_x
back_y
back_z
label
0
50000
100000
150000
200000
250000
300000
350000

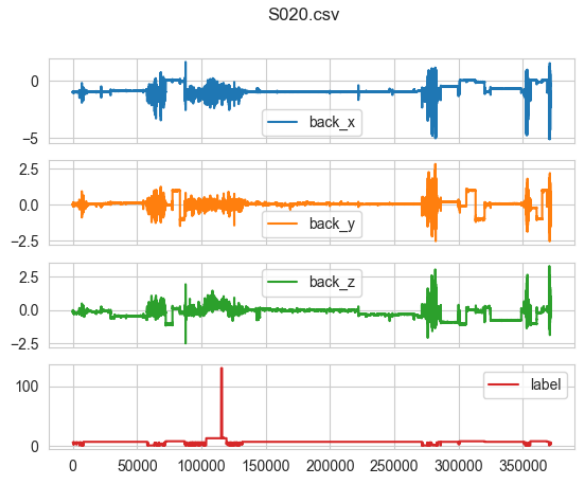

S017.csv
thigh_x
thigh_y
thigh_z
label
0
50000
100000
150000
200000
250000
300000
350000

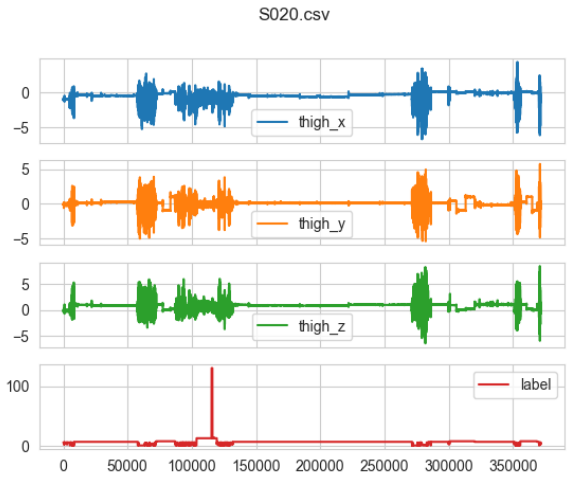

S018.csv
back_x
back_y
back_z
label
0
50000
100000
150000
200000
250000
300000

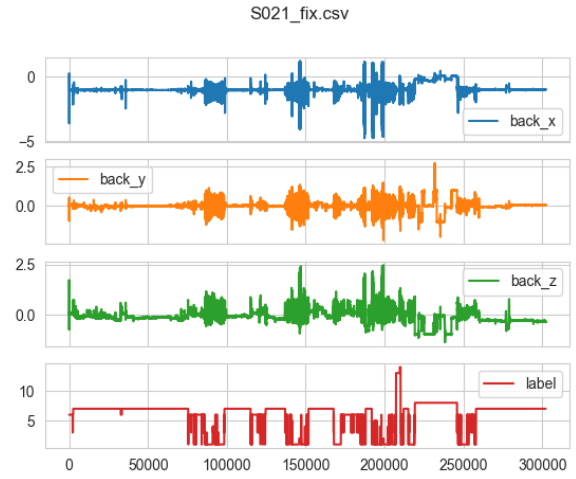

S018.csv
thigh_x
thigh_y
thigh_z
label
0
50000
100000
150000
200000
250000
300000

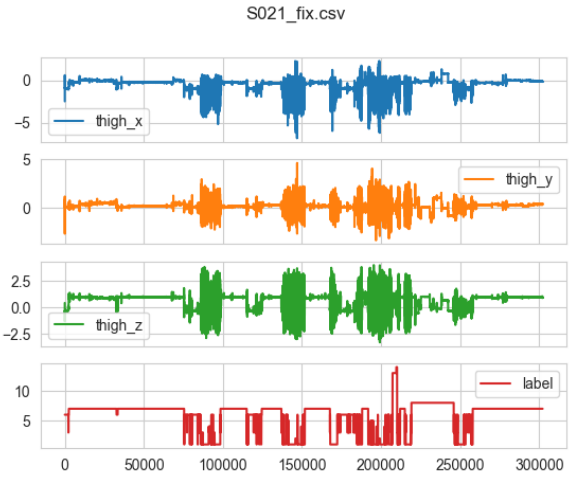

S019.csv
back_x
back_y
back_z
label
0
50000
100000
150000
200000
250000
300000

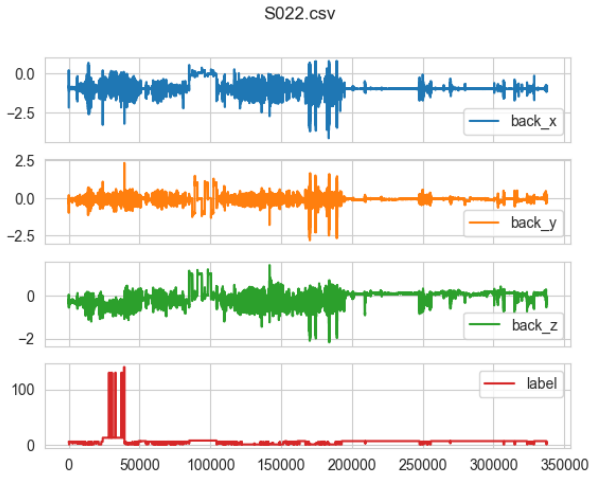

S019.csv
thigh_x
thigh_y
thigh_z
label
0
50000
100000
150000
200000
250000
300000

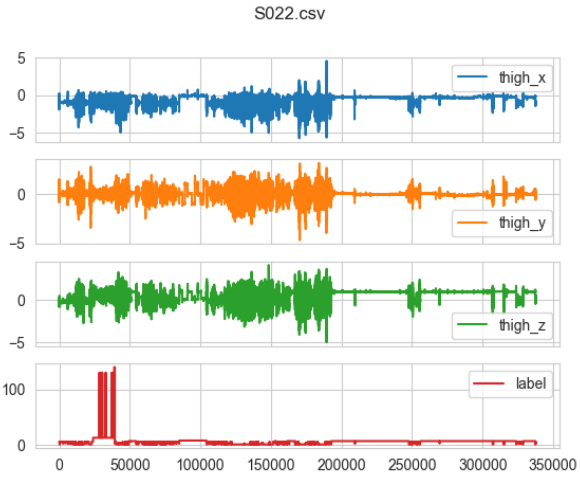

S020.csv
back_x
back_y
back_z
label


S020.csv
thigh_x
thigh_y
thigh_z
label

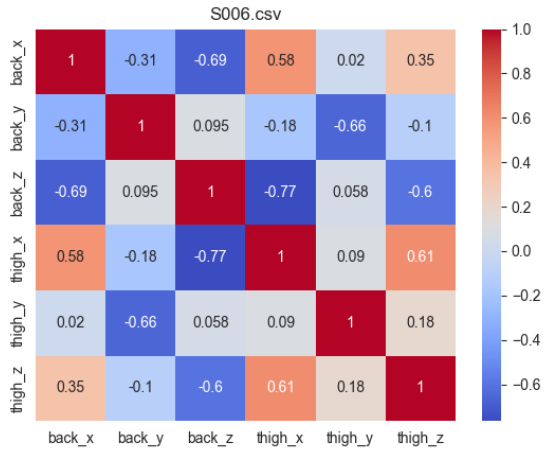

S021_fix.csv
back_x
back_y
back_z
label

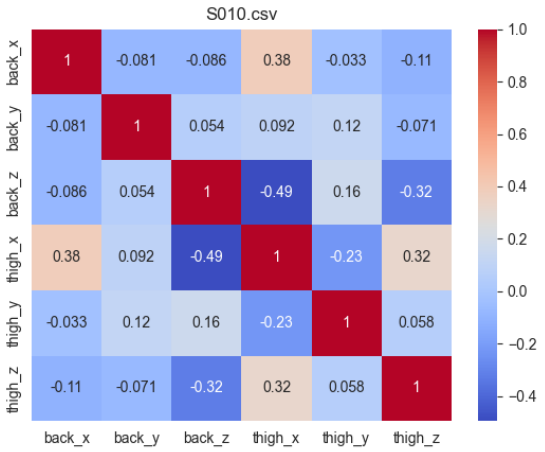

S021_fix.csv
thigh_x
thigh_y
thigh_z
label

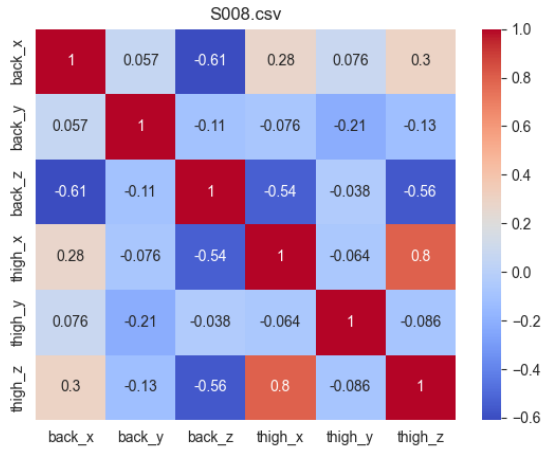

S022.csv
back_x
back_y
back_z
label

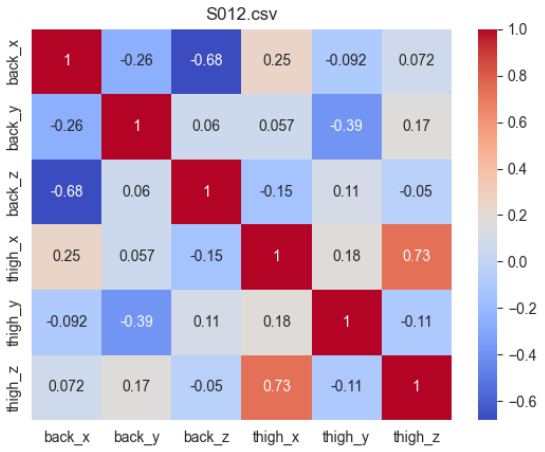

S022.csv
thigh_x
thigh_y
thigh_z
label

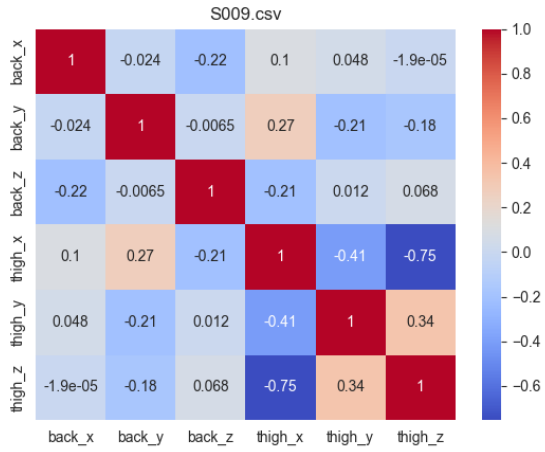
4.2 HEATMAPS

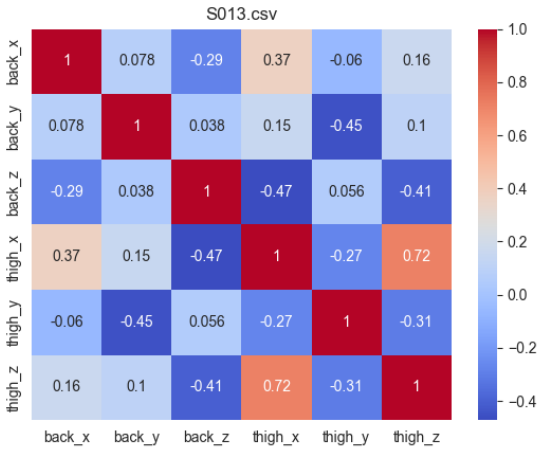

S006.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 -0.31 -0.69 0.58 0.02 0.35
-0.31 1 0.095 -0.18 -0.66 -0.1
-0.69 0.095 1 -0.77 0.058 -0.6
0.58 -0.18 -0.77 1 0.09 0.61
0.02 -0.66 0.058 0.09 1 0.18
0.35 -0.1 -0.6 0.61 0.18 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6


S010.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 -0.081 -0.086 0.38 -0.033 -0.11
-0.081 1 0.054 0.092 0.12 -0.071
-0.086 0.054 1 -0.49 0.16 -0.32
0.38 0.092 -0.49 1 -0.23 0.32
-0.033 0.12 0.16 -0.23 1 0.058
-0.11 -0.071 -0.32 0.32 0.058 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

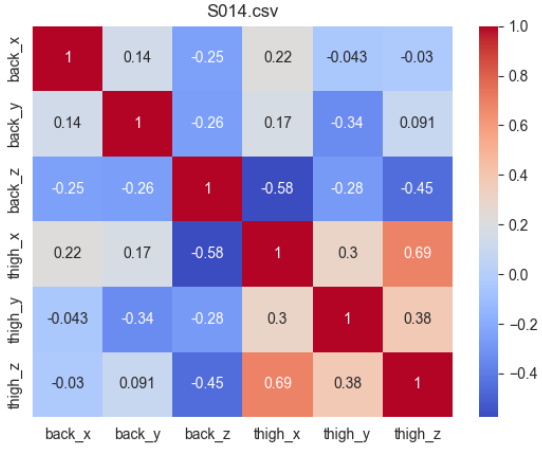

S008.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.057 -0.61 0.28 0.076 0.3
0.057 1 -0.11 -0.076 -0.21 -0.13
-0.61 -0.11 1 -0.54 -0.038 -0.56
0.28 -0.076 -0.54 1 -0.064 0.8
0.076 -0.21 -0.038 -0.064 1 -0.086
0.3 -0.13 -0.56 0.8 -0.086 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6


S012.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 -0.26 -0.68 0.25 -0.092 0.072
-0.26 1 0.06 0.057 -0.39 0.17
-0.68 0.06 1 -0.15 0.11 -0.05
0.25 0.057 -0.15 1 0.18 0.73
-0.092 -0.39 0.11 0.18 1 -0.11
0.072 0.17 -0.05 0.73 -0.11 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

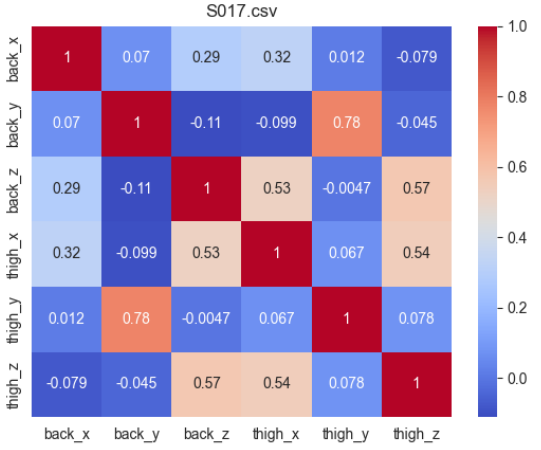

S009.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 -0.024 -0.22 0.1 0.048 -1.9e-05
-0.024 1 -0.0065 0.27 -0.21 -0.18
-0.22 -0.0065 1 -0.21 0.012 0.068
0.1 0.27 -0.21 1 -0.41 -0.75
0.048 -0.21 0.012 -0.41 1 0.34
-1.9e-05 -0.18 0.068 -0.75 0.34 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6


S013.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.078 -0.29 0.37 -0.06 0.16
0.078 1 0.038 0.15 -0.45 0.1
-0.29 0.038 1 -0.47 0.056 -0.41
0.37 0.15 -0.47 1 -0.27 0.72
-0.06 -0.45 0.056 -0.27 1 -0.31
0.16 0.1 -0.41 0.72 -0.31 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

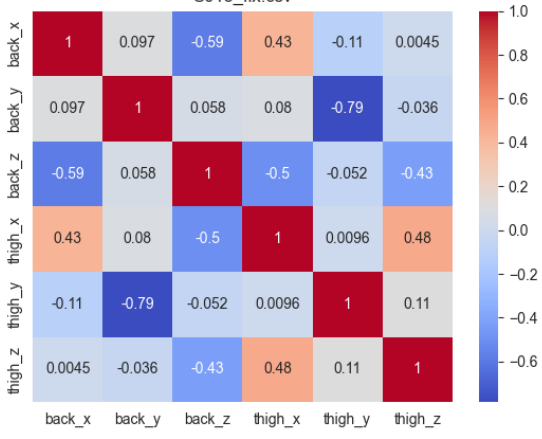
S014.csv



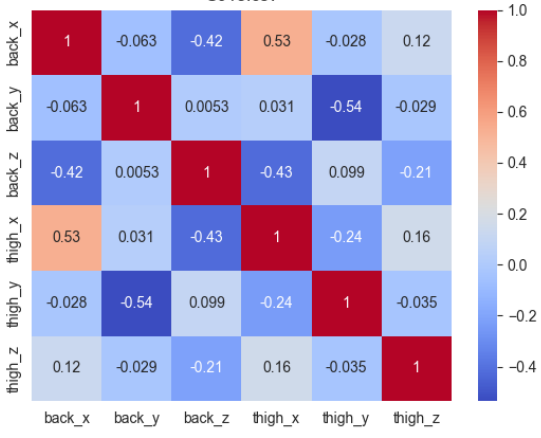
S017.csv



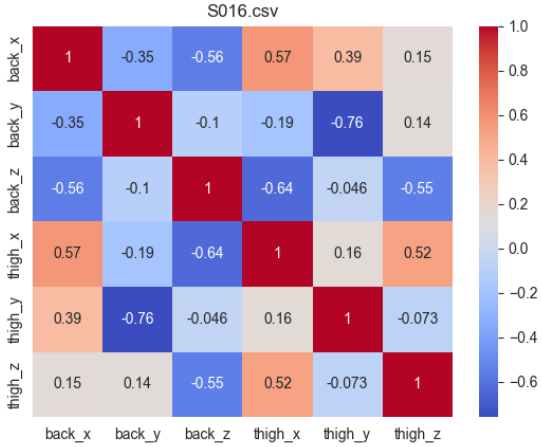
S015_fix.csv



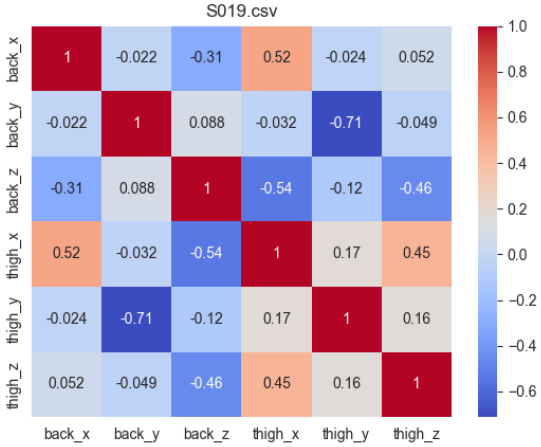
S018.csv

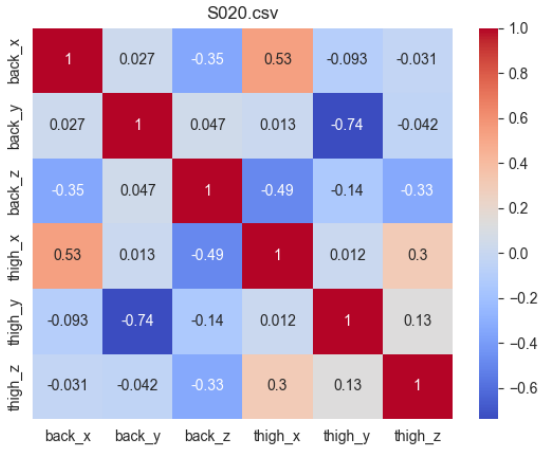


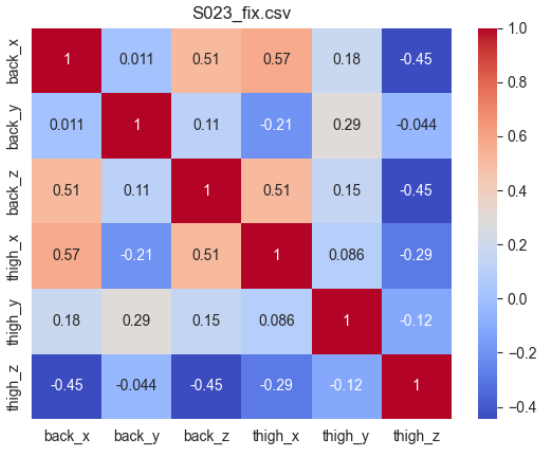
S016.csv

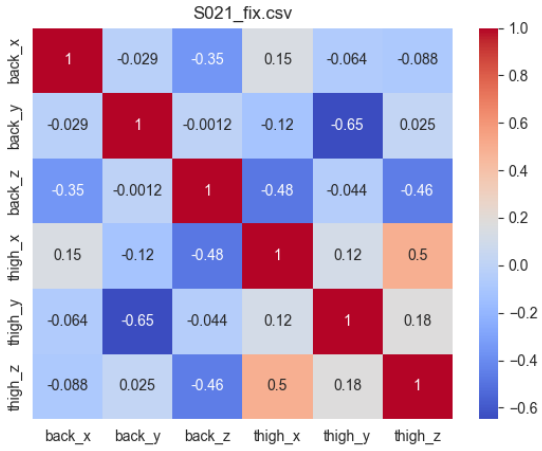


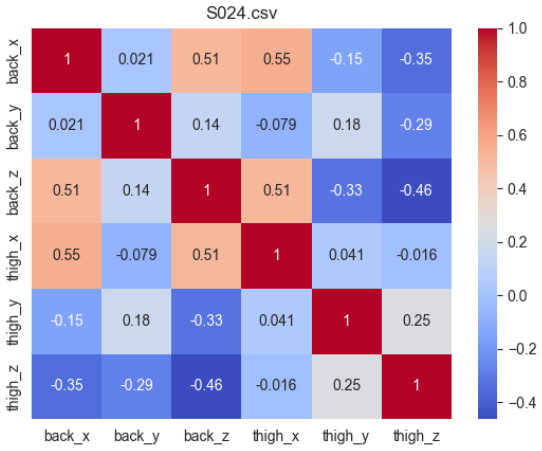
S019.csv

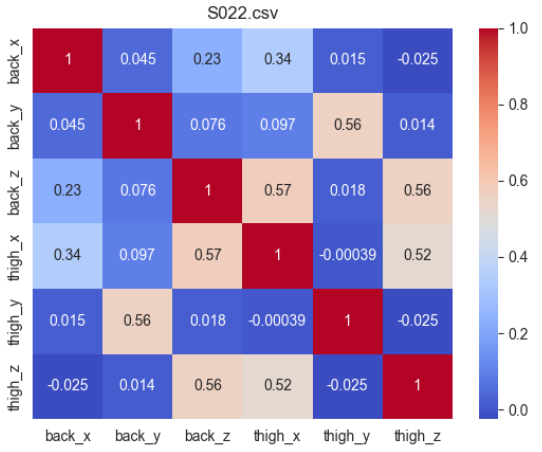


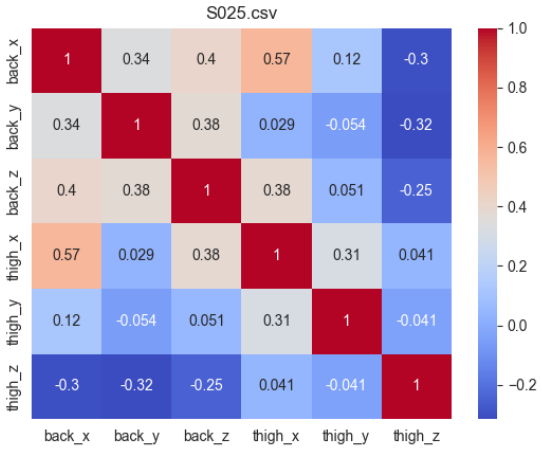

S020.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.027 -0.35 0.53 -0.093 -0.031
0.027 1 0.047 0.013 -0.74 -0.042
-0.35 0.047 1 -0.49 -0.14 -0.33
0.53 0.013 -0.49 1 0.012 0.3
-0.093 -0.74 -0.14 0.012 1 0.13
-0.031 -0.042 -0.33 0.3 0.13 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

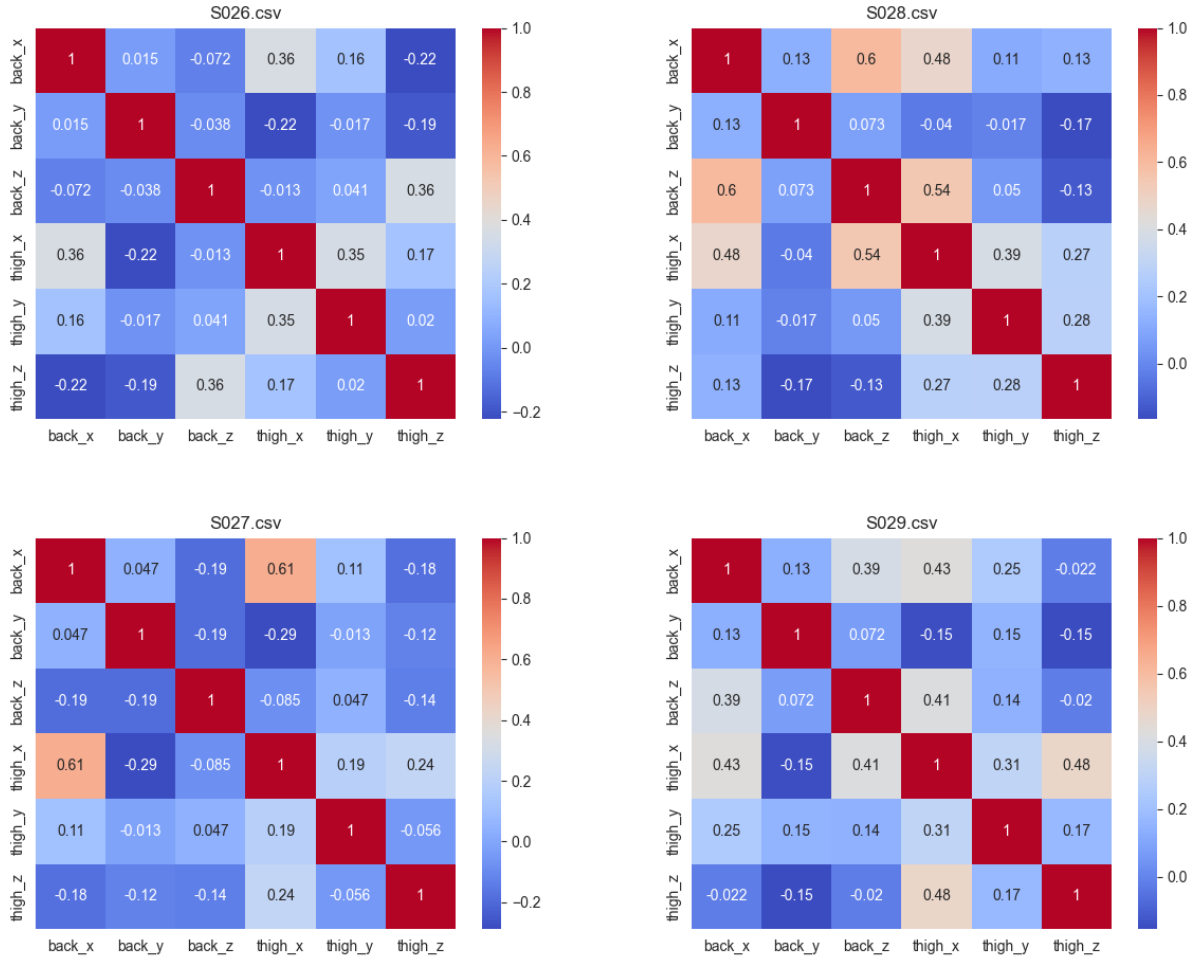

S023_fix.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.011 0.51 0.57 0.18 -0.45
0.011 1 0.11 -0.21 0.29 -0.044
0.51 0.11 1 0.51 0.15 -0.45
0.57 -0.21 0.51 1 0.086 -0.29
0.18 0.29 0.15 0.086 1 -0.12
-0.45 -0.044 -0.45 -0.29 -0.12 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


S021_fix.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 -0.029 -0.35 0.15 -0.064 -0.088
-0.029 1 -0.0012 -0.12 -0.65 0.025
-0.35 -0.0012 1 -0.48 -0.044 -0.46
0.15 -0.12 -0.48 1 0.12 0.5
-0.064 -0.65 -0.044 0.12 1 0.18
-0.088 0.025 -0.46 0.5 0.18 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6


S024.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.021 0.51 0.55 -0.15 -0.35
0.021 1 0.14 -0.079 0.18 -0.29
0.51 0.14 1 0.51 -0.33 -0.46
0.55 -0.079 0.51 1 0.041 -0.016
-0.15 0.18 -0.33 0.041 1 0.25
-0.35 -0.29 -0.46 -0.016 0.25 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


S022.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.045 0.23 0.34 0.015 -0.025
0.045 1 0.076 0.097 0.56 0.014
0.23 0.076 1 0.57 0.018 0.56
0.34 0.097 0.57 1 -0.00039 0.52
0.015 0.56 0.018 -0.00039 1 -0.025
-0.025 0.014 0.56 0.52 -0.025 1
1.0 0.8 0.6 0.4 0.2 0.0


S025.csv
back_x back_y back_z thigh_x thigh_y thigh_z
1 0.34 0.4 0.57 0.12 -0.3
0.34 1 0.38 0.029 -0.054 -0.32
0.4 0.38 1 0.38 0.051 -0.25
0.57 0.029 0.38 1 0.31 0.041
0.12 -0.054 0.051 0.31 1 -0.041
-0.3 -0.32 -0.25 0.041 -0.041 1
1.0 0.8 0.6 0.4 0.2 0.0 −0.2



4.3 ΚΩΔΙΚΑΣ

4.3.1 data_analysis.ipynb

```
import seaborn as sns
import pandas as pd
import os
import matplotlib.pyplot as plt

file_list = ['S006.csv','S008.csv','S009.csv','S010.csv','S012.csv','S013.csv','S014.csv','S015_fix.csv','S016.csv',
        'S017.csv','S018.csv','S019.csv','S020.csv','S021_fix.csv','S022.csv','S023_fix.csv','S024.csv','S025.csv',
        'S026.csv','S027.csv','S028.csv','S029.csv',]

# Append all df's into a single combined dataframe
df_combined = pd.DataFrame()

for file in file_list:
    df = pd.read_csv(os.path.join('harth/', file))
    df_combined = pd.concat([df_combined, df])

df_combined = df_combined.drop('label', axis=1)

# Αφαίρεση περιττών στηλών
df = pd.read_csv(os.path.join('harth/', file_list[7]))
df = df.drop('index', axis=1)
```

```
df.to_csv('harth/S015_fix.csv', index=False)

# Δημιουργία πινάκων
df = pd.read_csv(os.path.join('harth/', file_list[0]))
df.head()
df.info()
df_combined.describe()

# Δημιουργία γραφημάτων
for file in file_list:
    df = pd.read_csv(os.path.join('harth/', file_list[1]))
    df[['back_x','back_y', 'back_z', 'label']].plot(title= file, subplots=True)
    plt.show()
    df[['thigh_x','thigh_y', 'thigh_z', 'label']].plot(title= file, subplots=True)
    plt.show()

# Heatmaps
    for file in file_list:
        df = pd.read_csv(os.path.join('harth/', file))
        my_columns = ['back_x', 'back_y', 'back_z', 'thigh_x', 'thigh_y', 'thigh_z']
        df_selected = df[my_columns]

        corr_df = df_selected.corr()
        sns.heatmap(corr_df, annot=True, cmap='coolwarm')
        plt.title(file)

        # Αποθήκευση σε αρχείο:
        folder_path = 'Report/src/img/heatmaps'
        fig_name = file.replace(".csv","") + '_heatmap.png'
        plt.savefig(os.path.join(folder_path, fig_name))

        plt.show()
```

4.3.2 classifiers.ipynb

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import os
import pickle

file_list = ['S006.csv','S008.csv','S009.csv','S010.csv','S012.csv','S013.csv','S014.csv','S015_fix.
    csv','S016.csv',
         'S017.csv','S018.csv','S019.csv','S020.csv','S021_fix.csv','S022.csv','S023_fix.csv','S024.
    csv','S025.csv',
         'S026.csv','S027.csv','S028.csv','S029.csv',]

def get_classifier(option):
    if option == 1:
        classifier = MLPClassifier(max_iter=500)
    elif option == 2:
        classifier = RandomForestClassifier(n_estimators=100, criterion='entropy', random_state=42)
    elif option == 3:
        classifier = GaussianNB()
```

```
    else:
        raise ValueError('Invalid option')

    return classifier

# 1: Neural Network
# 2: Random Forest
# 3: Bayesian Network

models_train_acc = []
models_test_acc = []
models_precisions = []
models_recalls = []
models_f1s = []

for number in range(1, 4):
    option = number
    train_acc = []
    test_acc = []
    precisions = []
    recalls = []
    f1s = []
    for i, file in enumerate(file_list):
        df = pd.read_csv(os.path.join('harth/', file))
        df = df.drop('timestamp', axis = 1)

        X = df.drop(['label'], axis = 1)
        Y = df['label']

        X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

        clf = get_classifier(option)
        clf = clf.fit(X_train, Y_train)
        predictions = clf.predict(X_test)

        print(f"Classifier {option} yields training accuracy for file {file} of {clf.score(X_train,
    Y_train)} with a testing accuracy of {accuracy_score(Y_test, predictions)}")

        train_acc.append(clf.score(X_train,Y_train))
        test_acc.append(accuracy_score(Y_test, predictions))
        precisions.append(precision_score(Y_test, predictions, average='macro'))
        recalls.append(recall_score(Y_test, predictions, average='macro'))
        f1s.append(f1_score(Y_test, predictions, average='macro'))

    models_train_acc.append(np.mean(train_acc))
    models_test_acc.append(np.mean(test_acc))
    models_precisions.append(np.mean(precisions))
    models_recalls.append(np.mean(recalls))
    models_f1s.append(np.mean(f1s))

fig = plt.figure("Classification Results")
x_axis = np.arange(len(models_train_acc))
plt.bar(x_axis-0.2, models_train_acc, 0.4, label = "Train set")
plt.bar(x_axis+0.2, models_test_acc, 0.4, label = 'Test Set')
plt.xticks(x_axis)
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.legend()
plt.show()

fig = plt.figure("Classification Results")
```

```
x_axis = np.arange(len(models_train_acc))
plt.bar(x_axis-0.4, models_test_acc, 0.2, label = "Accuracy")
plt.bar(x_axis-0.2, models_precisions, 0.2, label = 'Precision')
plt.bar(x_axis, models_recalls, 0.2, label = 'Recall')
plt.bar(x_axis+0.2, models_f1s, 0.2, label = 'F1')
plt.xticks(x_axis)
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

4.3.3 clustering.ipynb

```
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

file_list = ['S006.csv','S008.csv','S009.csv','S010.csv','S012.csv','S013.csv','S014.csv','S015_fix.csv','S016.csv',
     'S017.csv','S018.csv','S019.csv','S020.csv','S021_fix.csv','S022.csv','S023_fix.csv','S024.csv','S025.csv',
     'S026.csv','S027.csv','S028.csv','S029.csv',]

df_combined = pd.DataFrame()

for file in file_list:
    df = pd.read_csv(os.path.join('harth/', file))
    df_combined = pd.concat([df_combined, df])

df_combined = df_combined.drop('label', axis=1)

data = df_combined
data.drop(['timestamp'], axis=1, inplace=True)

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(data_scaled)

pca = PCA(n_components=2)
principal_components = pca.fit_transform(data_scaled)

plt.figure()
plt.scatter(principal_components[:, 0], principal_components[:, 1], c=clusters, cmap='plasma', alpha=0.5)
plt.title('KMeans Clustering')
plt.colorbar()
plt.show()

df = pd.read_csv(os.path.join('harth/', file_list[7]))
df = df.drop(['timestamp', 'label'], axis = 1)

scaler = StandardScaler()
data_scaled = scaler.fit_transform(df.head(10000))
```

```
45
46 dbscan = DBSCAN(eps=1, min_samples=10)
47 clusters = dbscan.fit_predict(data_scaled)
48
49 plt.scatter(data_scaled[:, 0], data_scaled[:, 1], c=clusters, cmap='plasma', alpha=0.5)
50 plt.title('DBSCAN Clustering')
51 # plt.colorbar()
52 plt.show()
```