Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

Project

Ζήκος Σπυρίδων, 1084581, 4ο έτος

Contents

Περιβάλλον Υλοποίησης

Έκδοση python: 3.7.9

Έκδοση pip: 24.0

Βιβλιοθήκες: seaborn, matplotlib, pandas, sklearn, time

Κατεβάζουμε την python και την εγκαθιστούμε από την επίσημη ιστοσελίδα.

Εγκαθιστούμε την βιβλιοθήκη X εκτελόντας την εντολή: `pip install X` (στο command line)

Διαδικασία Υλοποίησης

Ερώτημα 1

Αρχικά, διαβάζουμε τα δεδομένα από τα αρχεία excel και τα αποθηκεύουμε και σε ένα ενιαίο dataframe αλλά και σε λίστα όπου κάθε στοιχείο είναι ένα dataframe που αντιστοιχεί σε κάποιον συμμετέχοντα. Έπειτα, διαγράφουμε τις στήλες που δεν χρειαζόμαστε και εκτυπώνουμε κάποια στατιστικά στοιχεία για τα δεδομένα. Επίσης, φτιάχνουμε διαγράμματα με την κατανομή που ακολουθούν τα χαρακτηριστικά και την συσχέτιση που έχουν μεταξύ τους.

Ερώτημα 2

Για να εκπαιδεύσουμε τους ταξινομητές και να προβάλουμε τα αποτελέσματα χρησιμοποιούμε την συνάρτηση classifiers() η οποία μετατρέπει τα δεδομένα εισόδου ώστε να περιλαμβάνουν μετρήσεις από τρεις διαδοχικές χρονικές στιγμές και την ετικέτα της τελευταίας μέτρησης. Ύστερα, χωρίζουμε τα δεδομένα σε training(70%) και testing(30%) και εκπαιδεύουμε τους τρεις ταξινομητές με τα δεδομένα training. Ύστερα, εκτυπώνουμε διάφορες μετρικές που προκύπτουν από την απόδοση των μοντέλων στα δεδομένα testing.

Ερώτημα 3

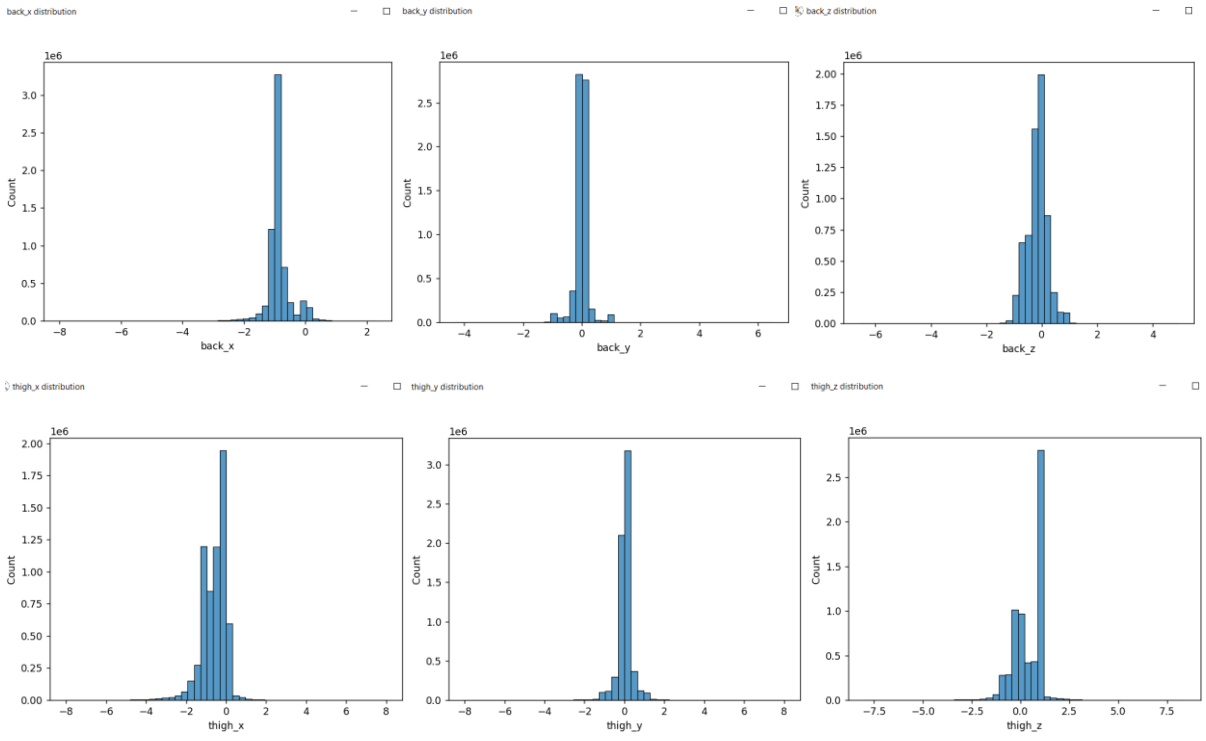
Υποθέσαμε ότι η ομαδοποίηση είναι unsupervised και δεν έχουμε πρόσβαση σε μία αντικειμενική ομαδοποίηση. Για να κάνουμε την ομαδοποίηση των συμμετεχόντων σε συστάδες και να προβάλουμε τα αποτελέσματα χρησιμοποιούμε την συνάρτηση clusterers() η οποία χρησιμοποιεί τον μέσο όρο, την διασπορά, το μέγιστο και το ελάχιστο του κάθε χαρακτηριστικού του κάθε συμμετέχοντα. Έτσι, εκτελούμε τους τρεις αλγορίθμους ομαδοποίησης και τυπώνουμε τα αποτελέσματα και τις μετρικές που μας δίνουν.

Σχολιασμός Αποτελεσμάτων

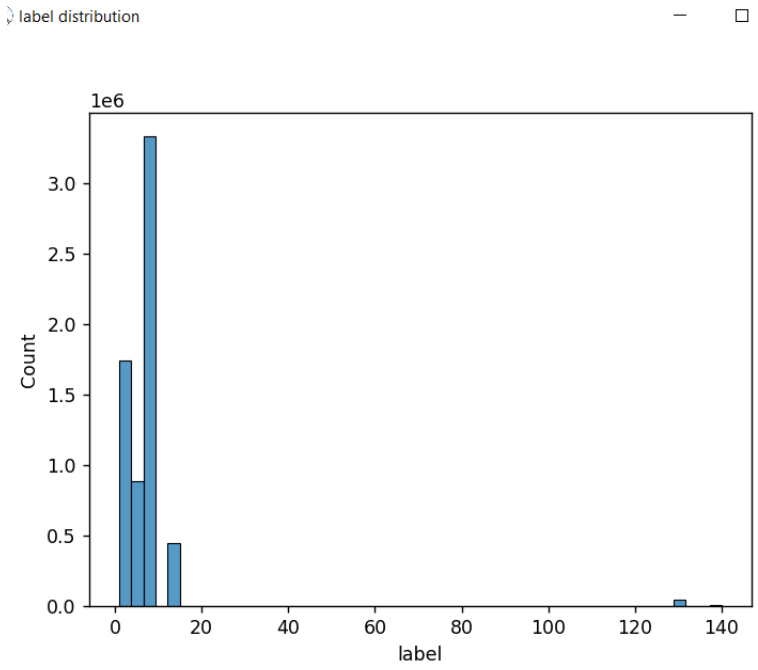
Ερώτημα 1

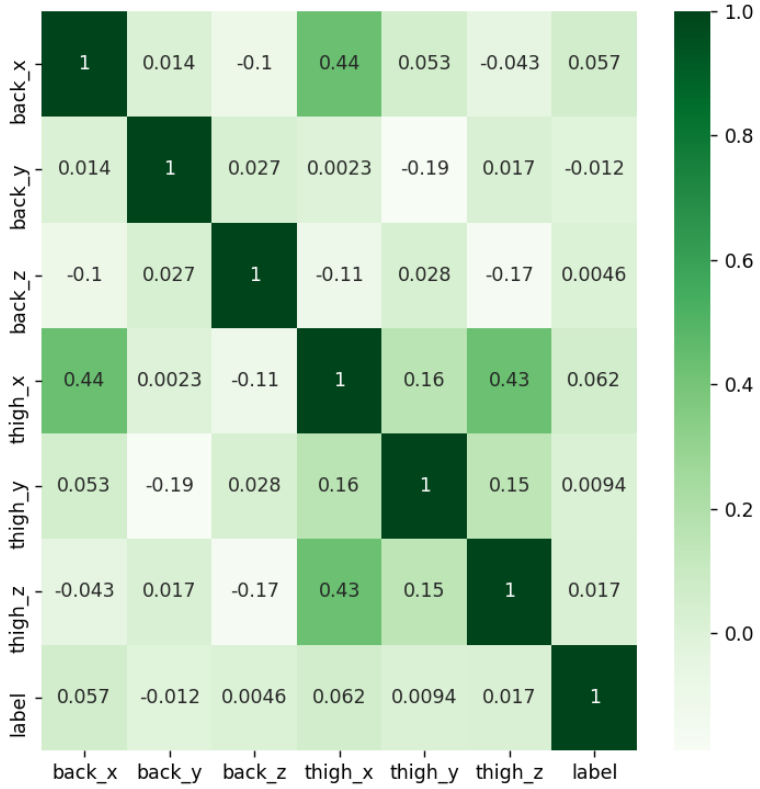
```
RangeIndex: 6461328 entries, 0 to 6461327
Data columns (total 8 columns):
 #   Column     Dtype
---  ------     -----
 0   timestamp  object
 1   back_x     float64
 2   back_y     float64
 3   back_z     float64
 4   thigh_x    float64
 5   thigh_y    float64
 6   thigh_z    float64
 7   label      int64
dtypes: float64(6), int64(1), object(1)
memory usage: 394.4+ MB
```

	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
count	6.461328e+06	6.461328e+06	6.461328e+06	6.461328e+06	6.461328e+06	6.461328e+06	6.461328e+06
mean	-8.849574e-01	-1.326128e-02	-1.693779e-01	-5.948883e-01	2.087665e-02	3.749160e-01	6.783833e+00
std	3.775916e-01	2.311709e-01	3.647385e-01	6.263466e-01	3.884511e-01	7.360983e-01	1.143238e+01
min	-8.000000e+00	-4.307617e+00	-6.574463e+00	-8.000000e+00	-7.997314e+00	-8.000000e+00	1.000000e+00
25%	-1.002393e+00	-8.312914e-02	-3.720700e-01	-9.742110e-01	-1.000873e-01	-1.557138e-01	3.000000e+00
50%	-9.748998e-01	2.593677e-03	-1.374510e-01	-4.217309e-01	3.262909e-02	7.004390e-01	7.000000e+00
75%	-8.123032e-01	7.251000e-02	4.647321e-02	-1.678755e-01	1.549512e-01	9.486747e-01	7.000000e+00
max	2.291708e+00	6.491943e+00	4.909483e+00	7.999756e+00	7.999756e+00	8.406235e+00	1.400000e+02



Παρατηρούμε ότι η κατανομή των χαρακτηριστικών μοιάζει με κανονική κατανομή.





Παρατηρούμε ότι τα χαρακτηριστικά back_x,thing_x και thing_x,thing_z είναι αρκετά συσχετισμένα μεταξύ τους.

Ερώτημα 2

```
Classifier: Multi-layer Perceptron
        Training accuracy: 0.9020194440415589
        Testing accuracy: 0.9017549543489005

              precision    recall  f1-score   support

           1       0.81      0.90      0.85    359321
           2       0.96      0.94      0.95     87539
           3       0.48      0.16      0.24     76634
           4       0.58      0.17      0.26     22966
           5       0.56      0.15      0.24     20076
           6       0.78      0.91      0.84    222046
           7       0.99      1.00      1.00    871837
           8       1.00      1.00      1.00    128328
          13       0.86      0.91      0.88    117876
          14       0.76      0.62      0.68     16768
         130       0.55      0.56      0.55     12619
         140       0.54      0.45      0.49      2388

    accuracy                           0.90   1938398
   macro avg       0.74      0.65      0.67   1938398
weighted avg       0.89      0.90      0.89   1938398

        Elapsed time:  2786.3669633865356
```

```
Classifier: Random Forest
        Training accuracy: 0.9999960202771302
        Testing accuracy: 0.927286346766763

              precision    recall  f1-score   support

           1       0.83      0.93      0.88    359321
           2       0.97      0.97      0.97     87539
           3       0.61      0.37      0.46     76634
           4       0.86      0.47      0.61     22966
           5       0.83      0.31      0.45     20076
           6       0.87      0.90      0.89    222046
           7       1.00      1.00      1.00    871837
           8       1.00      1.00      1.00    128328
          13       0.89      0.95      0.92    117876
          14       0.87      0.73      0.80     16768
         130       0.79      0.63      0.70     12619
         140       0.79      0.59      0.67      2388

    accuracy                           0.93   1938398
   macro avg       0.86      0.74      0.78   1938398
weighted avg       0.92      0.93      0.92   1938398

        Elapsed time:  9780.46250796318
```

```
Classifier: Naive Bayes
        Training accuracy: 0.7581880587088717
        Testing accuracy: 0.758255012644462

              precision    recall  f1-score   support

           1       0.74      0.38      0.50    359321
           2       0.61      0.76      0.68     87539
           3       0.16      0.19      0.17     76634
           4       0.08      0.04      0.06     22966
           5       0.19      0.02      0.03     20076
           6       0.61      0.91      0.73    222046
           7       0.98      0.97      0.98    871837
           8       0.96      0.98      0.97    128328
          13       0.53      0.52      0.53    117876
          14       0.33      0.46      0.38     16768
         130       0.09      0.29      0.14     12619
         140       0.04      0.66      0.08      2388

    accuracy                           0.76   1938398
   macro avg       0.44      0.52      0.44   1938398
weighted avg       0.79      0.76      0.76   1938398

        Elapsed time:  42.77476215362549
```

Παρατηρούμε ότι ο ταξινομητής Random Forest έχει την καλύτερη επίδοση σε όλες τις μετρικές αξιολόγησης αλλά κάνει τον περισσότερο χρόνο για να τερματίσει. Αντιθέτως, ο Naïve Bayes αν και έχει την χειρότερη επίδοση σε όλες τις μετρικές αξιολόγησης, τερματίζει πολύ γρήγορα. Τέλος, ο ταξινομητής Multi-Layer Perceptron έχει λίγο χειρότερη απόδοση από τον Random Forest αλλά τερματίζει σχεδόν στο 1/4 του χρόνου.

Ερώτημα 3

```
Clusterer: KMeans
        Number of clusters: 3
        Number of noise points: 0
        Silhouette Coefficient: 0.254
        Labels: [2 2 2 2 2 0 0 2 0 2 0 2 0 2 2 0 1 1 2 0 0 1]
        Elapsed time:  0.010984420776367188
Clusterer: DBSCAN
        Number of clusters: 2
        Number of noise points: 5
        Silhouette Coefficient: 0.241
        Labels: [ 0  0  0  0  0  1  1  0  1  0  1  0  1  0  0 -1 -1 -1  0 -1  0 -1]
        Elapsed time:  0.0029997825622558594
Clusterer: Birch
        Number of clusters: 3
        Number of noise points: 0
        Silhouette Coefficient: 0.269
        Labels: [0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 1 2 1 0 1 0 2]
        Elapsed time:  0.0030028820037841797
```

Παρατηρούμε ότι και οι 3 αλγόριθμοι τοποθετούν τους 15 πρώτους συμμετέχοντες στις ίδιες ομάδες. Ο DBSCAN και ο Birch έχουν βάλει διαφορετικό label μόνο σε 5 συμμετέχοντες. Ο Kmeans με τον Birch έχουν βάλει διαφορετικό label μόνο σε 2 συμμετέχοντες.

Επίσης, έχουν παραπλήσιο silhouette coefficient με τον Birch να έχει τον μεγαλύτερο και τον DBSCAN τον μικρότερο.

Σύμφωνα με την Wikipedia:

> A clustering with an average silhouette width of over 0.7 is considered to be "strong", a value over 0.5 "reasonable" and over 0.25 "weak", but with increasing dimensionality of the data, it becomes difficult to achieve such high values because of the curse of dimensionality, as the distances become more similar.

Οπότε οι αλγόριθμοι συσταδοποίησης που χρησιμοποιήσαμε επιδέχονται βελτιστοποίηση.