ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ ΜΑΘΗΣΗΣ

Όνομα	ΧΡΙΣΤΙΝΑ-ΕΛΕΑΝΝΑ
Επώνυμο	ΣΑΜΑΡΑ
Έτος	4ο
ΑΜ	1084622

Ημερομηνία Παράδοσης: 7/06/2024

Περιβάλλον Υλοποίησης

Το project υλοποιήθηκε σε python, χρησιμοποιώντας τις βιβλιοθήκες: **pandas, numpy, scikit-learn, matplotlib, seaborn, glob, os, pgmpy**. Τα τρία μοντέλα εκπαίδευσης υλοποιήθηκαν ως: RandomForest από την sklearn.enseble, NeuralNetwork από την sklearn.neural_network και BayesianNetwork από την pgmpy.models. Οι δύο αλγόριθμοι ομαδοποίησης KMeans και MiniBatchKMeans υλοποιήθηκαν από την sklearn.cluster.

Πρώτη Ανάλυση Δεδομένων

Με την συνάρτηση **read_dataset()** φορτώνουμε όλα τα δεδομένα από τα αρχεία του **harth** στο ίδιο dataframe. Μετατρέπουμε την χρονική στήλη timestamp σε time_diff, που εκφράζει την διαφορά μεταξύ της τρέχουσας εγγραφής και της προηγούμενής της.

Στην συνέχεια, υπολογίζουμε τα βασικά στατιστικά μεγέθη των διαφορετικών χαρακτηριστικών.

Έχουμε:

Μέσοι Όροι

label	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	time_diff
1	-0.992566	-0.038755	-0.137808	-1.056683	-0.023477	-0.074345	0.000143
2	-0.965280	-0.076626	-0.259829	-1.246811	-0.164790	-0.140530	0.000000
3	-0.982356	-0.022316	-0.031349	-0.974374	0.020759	-0.068220	0.000705
4	-0.961095	-0.033405	-0.065052	-1.016784	0.013345	-0.006504	0.000093
5	-0.994845	-0.034358	-0.082835	-1.040411	0.000524	-0.028122	0.000140
6	-0.985741	-0.000301	-0.046233	-0.962695	0.017709	-0.122115	0.001280
7	-0.919736	0.015989	-0.234789	-0.220810	0.062572	0.900163	0.000785
8	-0.027791	-0.050431	-0.141276	-0.040130	0.047339	0.127901	0.000000
13	-0.900350	-0.054708	-0.098196	-0.876525	-0.014650	0.008914	0.000000
14	-0.822610	-0.108429	-0.261261	-1.037931	-0.071229	-0.071526	0.000000
130	-0.856316	-0.054885	-0.044627	-0.851349	-0.002741	-0.091765	0.000000
140	-0.901097	-0.137201	-0.455977	-0.888563	-0.072356	-0.061449	0.000000

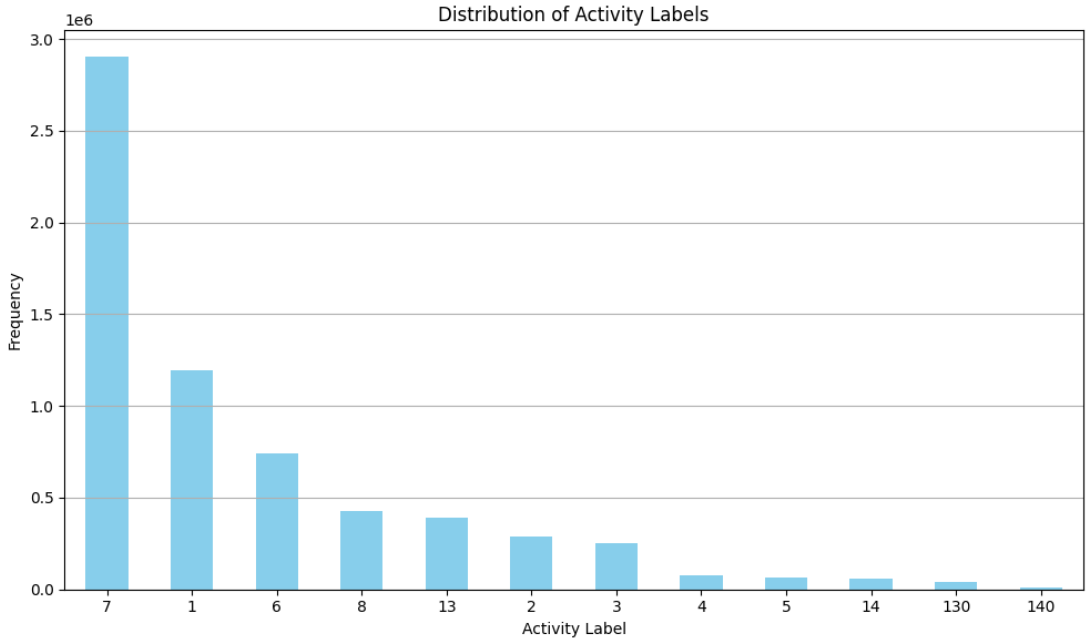
Τυπική Απόκλιση

label	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	time_diff
1	0.311378	0.190476	0.287737	0.639900	0.536738	0.721997	0.002767
2	1.113858	0.407701	0.451772	1.438550	0.898353	1.381836	0.000000
3	0.106649	0.158300	0.208112	0.192159	0.226300	0.294548	0.006877
4	0.268016	0.170207	0.336750	0.496791	0.308519	0.455921	0.001360
5	0.378185	0.209421	0.248571	0.531531	0.459857	0.623766	0.001666
6	0.069339	0.103504	0.190384	0.108487	0.125110	0.233771	0.010937
7	0.132376	0.087736	0.314371	0.198430	0.164567	0.282412	0.004417
8	0.130318	0.670548	0.674811	0.183538	0.710983	0.689791	0.000000
13	0.169175	0.132258	0.494732	0.530159	0.268370	0.834474	0.000000
14	0.320789	0.285032	0.599131	0.524064	0.289866	0.479070	0.000000
130	0.146399	0.115497	0.566928	0.286887	0.178055	0.593241	0.000000
140	0.111606	0.134170	0.272238	0.214518	0.133333	0.529570	0.000000

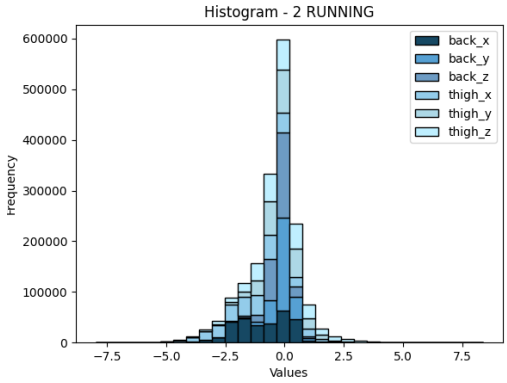
Επιπλέον, με την συνάρτηση **info()** εμφανίζουμε πληροφορίες σχετικά με το dataframe:

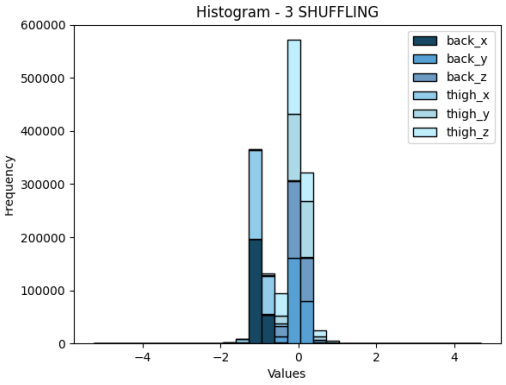
```
RangeIndex: 6461328 entries, 0 to 6461327
Data columns (total 8 columns):
 #   Column     Dtype
---  ------     -----
 0   back_x     float64
 1   back_y     float64
 2   back_z     float64
 3   thigh_x    float64
 4   thigh_y    float64
 5   thigh_z    float64
 6   label      int64
 7   time_diff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

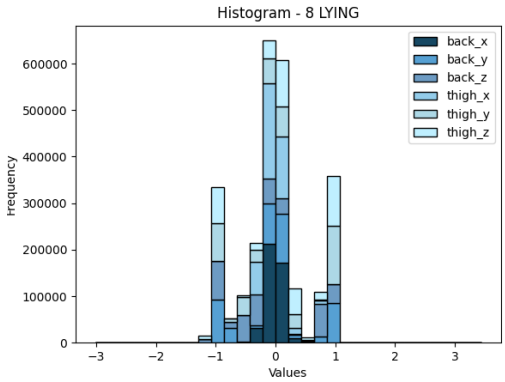
Στο παρακάτω γράφημα παρουσιάζεται η συχνότητα εγγραφών του dataset ανάλογα με το label

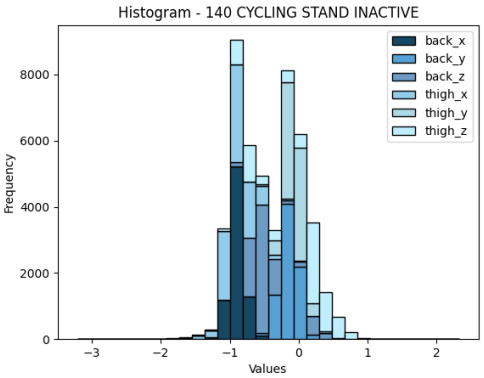


Βλέπουμε ότι η πολυπληθέστερη κλάση είναι η **7**, ενώ οι κλάσεις **4**, **5**, **14**, **130** και **140** έχουν λίγες εγγραφές. Γνωρίζοντας αυτό, μπορούμε να περιμένουμε καλύτερη απόδοση στην πρόβλεψη της κλάσης 7 σε σύγκριση με τις υπόλοιπες. Επίσης, παρατίθενται ιστογράμματα ορισμένων label (τα πιο χαρακτηριστικά):

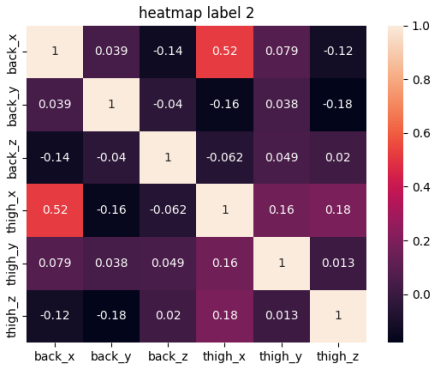


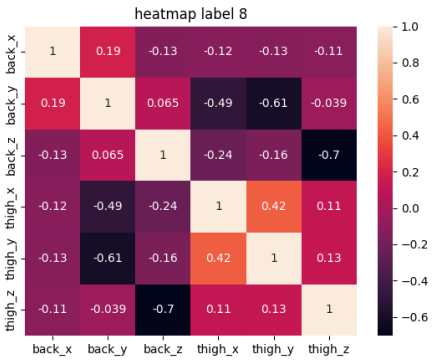






Παρατηρούμε ότι οι κατανομές των διαφόρων μεταβλητών ανά ετικέτα διαφοροποιούνται. Επίσης, έχουν υλοποιηθεί γραφήματα συσχέτισης (correlation heatmaps). Παρακάτω βλέπουμε τα γραφήματα συσχέτισης για τις ετικέτες 2 και 8. Σε αυτά διακρίνουμε μέτρια θετική συσχέτιση (0.52) μεταξύ **back_x** και **thigh_x** για την ετικέτα 2, και μια σχετικά δυνατή αρνητική συσχέτιση (-0.61) μεταξύ **back_y** και **thigh_z** για την ετικέτα 8.





Αρχείο για γραφήματα: **1_graphs.py**

Εκπαίδευση Μοντέλων

Τα δεδομένα χωρίστηκαν σε σύνολα εκπαίδευσης και ελέγχου ως 80% - 20%

Neural Networks

Για την εκπαίδευση ενός Neural Network μοντέλου χρησιμοποιήσαμε ένα MLP Classifier με τέσσερα (128, 64, 32, 64) κρυφά επίπεδα, καθώς και την ενεργοποίηση relu. Ο μέγιστος αριθμός επαναλήψεων είναι 10.

Random Forest

Για την εκπαίδευση του μοντέλου Random Forest χρησιμοποιήσαμε την συνάρτηση RandomForestClassifier της sklearn με παραμέτρους: μέγιστο βάθος 30, ελάχιστον πλήθος φύλλων 1, ελάχιστο πλήθος δειγμάτων για διαίρεση 2 και αριθμός δέντρων 50. Οι τιμές των max_depth, min_samples_leaf, min_samples_split, και n_estimators βρέθηκαν με τον αλγόριθμο GridSearchCV για την βελτιστοποίηση των παραμέτρων. Ορίσαμε ως μέγιστο βάθος το 30 (έναντι του None που βρέθηκε στην βελτιστοποίηση) για την μείωση του χρόνου εκτέλεσης.

Bayesian Networks

Αρχικά χρησιμοποιήθηκε η συνάρτηση qcut(), με την οποία το σύνολο δεδομένων χωρίστηκε σε 20 ίσα διαστήματα (bins) και διακριτοποιήθηκε. Στην συνέχεια, εκτελούμε τον αλγόριθμο hillClimbing, ο οποίος χρησιμοποιείται για την εύρεση της πιθανής δομής του δικτύου Bayes. Ο αλγόριθμος επιστρέφει τις ακμές του δικτύου και προστίθενται στο μοντέλο μας.

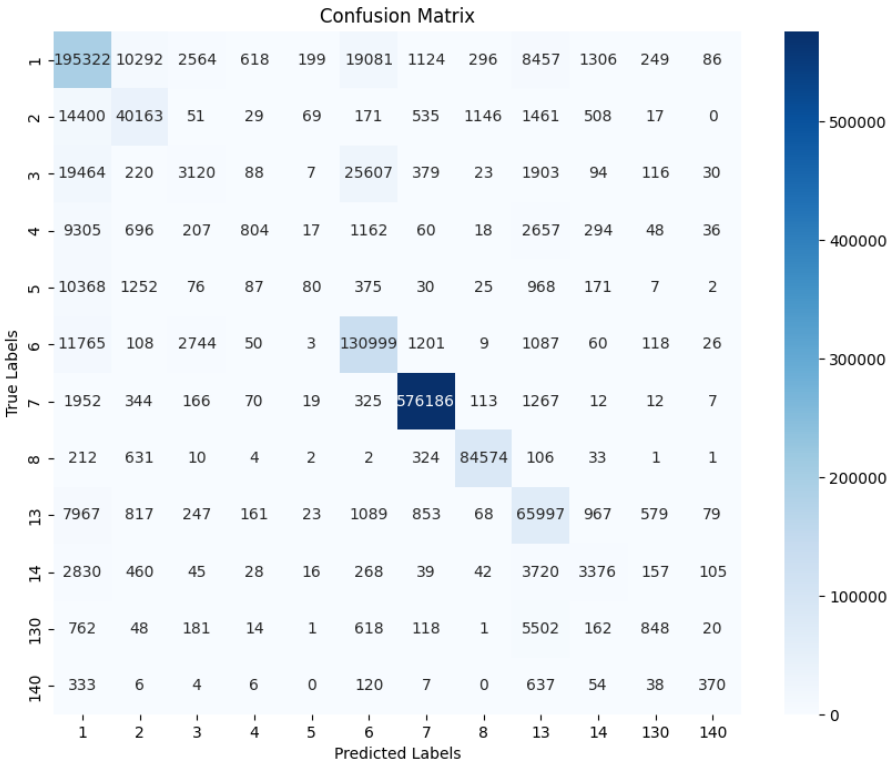
Μετρικές Αξιολόγησης

Για την αξιολόγηση και σύγκριση των μοντέλων αποθηκεύσαμε σε csv αρχεία τις μεταβλητές τους **y_test** και **y_predictions**. Τα αρχεία αυτά μετατρέπονται σε πίνακες numpy και χρησιμοποιήθηκαν για την δημιουργία λεξικών για τα **True Positives TP**, **True Negatives TN**, **False Positives FP** και **False Negatives FN**. Τα λεξικά αυτά περιέχουν τις αντίστοιχες τιμές για κάθε κλάση ξεχωριστά.

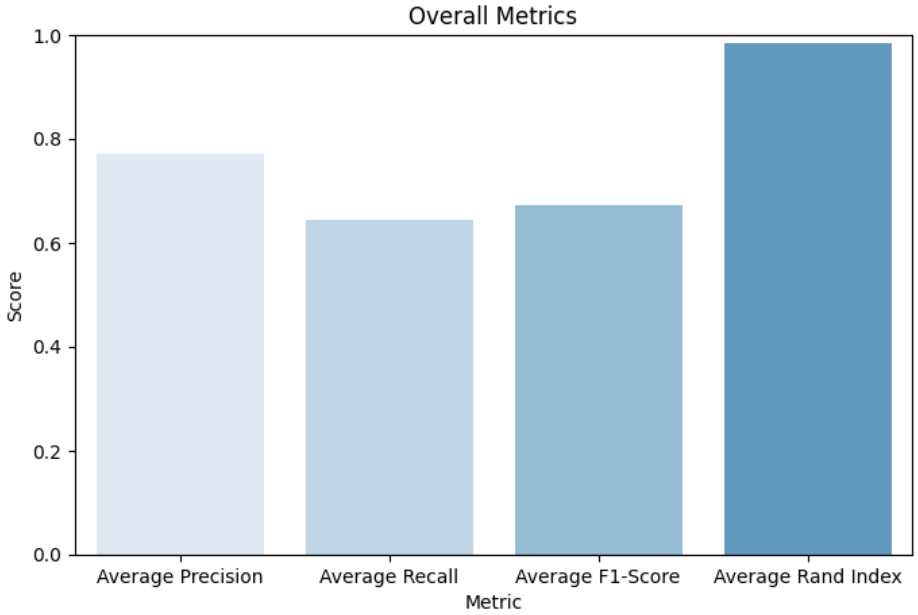
Στην συνέχεια υπολογίζουμε ανάκτηση, ακρίβεια, f1-score και rand index για κάθε κλάση, καθώς και τους μέσους όρους αυτών για κάθε μέθοδο.

Για την καλύτερη απεικόνιση των αποτελεσμάτων δημιουργήθηκαν γραφήματα.

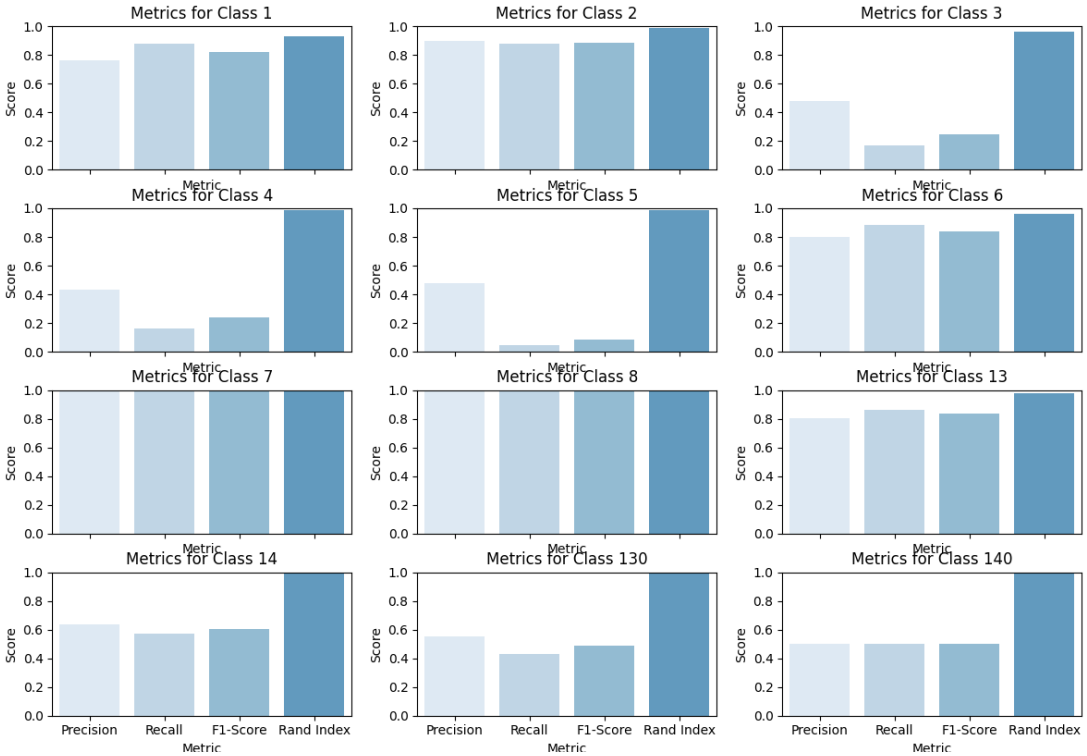
Για κάθε μοντέλο δημιουργήθηκε confusion matrix το οποίο απεικονίζει τις προβλέψεις των μοντέλων, δηλαδή πόσες από τις εγγραφές προβλέφθηκαν σωστά και πόσες λανθασμένα. Παρακάτω έχουμε το confusion matrix του μοντέλου Bayesian Networks. Όσο πιο έντονο χρώμα έχουν τα τετράγωνα τόσο πιο πολλές εγγραφές πέφτουν στην συγκεκριμένη κατηγορία. Βλέπουμε ότι το χρώμα στο (7, 7) είναι το πιο έντονο, κάτι το οποίοπεριμέναμε από την προεπεξεργασία, καθώς λόγω των πολλών εγγραφών με label 7 η εκπαίδευση για το συγκεκριμένο έγινε καλύτερα και είχε περισσότερες πιθανότητες. Το συγκεκριμένο γράφημα θα μπορούσε να χρησιμοποιηθεί στην ενοποίηση ορισμένων κλάσεων, έτσι ώστε να έχουμε καλύτερη απόδοση. Για παράδειγμα, η κλάση 130 προβλέφθηκε ως κλάση 13 πολλές φορές με διαφορά.



Ένα άλλο γράφημα που δημιουργήθηκε είναι το barplot που απεικονίζει τους μέσους όρους των μετρικών κάθε μοντέλου. Παρακάτω έχουμε το bar plot του μοντέλου Random Forest. Όπως βλέπουμε ο Random Forest έχει σχετικά καλή απόδοση, με όλες τις μετρικές του να είναι άνω του 0,6.



Τελευταίο γράφημα είναι τα bar plots των μετρικών για κάθε κλάση. Παρακάτω βλέπουμε το γράφημα του μοντέλου Neural Networks. Αυτό που είναι σημαντικό να παρατηρήσουμε είναι το πόσο χαμηλή απόδοση έχει το μοντέλο για ορισμένες κλάσεις. Αυτές είναι οι 3, 4 και 5, ενώ ακολουθούν και οι 130, 140. Οι κλάσεις αυτές είχαν το μικρότερο πλήθος δειγμάτων στο σύνολο δεδομένων μας. Αντίθετα, οι κλάσεις 7 και 8, από τις πιο πολυπληθείς, έχουν υψηλές αποδόσεις.



Όλα τα γραφήματα υπάρχουν στον φάκελο παράδοσης.

Αποτελέσματα Bayesian

```
Average Recall: 0.49675227717854614
Average Precision: 0.5935691614854659
Average F1-Score: 0.505326046758172
Average Rand Index: 0.9754401699546894
```

Αποτελέσματα Neural Networks

```
Average Recall: 0.6153818467592943
Average Precision: 0.6954878482204009
Average F1-Score: 0.6279085708635076
Average Rand Index: 0.9814402246389934
```

Αποτελέσματα Random Forest

```
Average Recall: 0.6432261454472242
Average Precision: 0.7717752822645543
Average F1-Score: 0.6740152078791596
Average Rand Index: 0.9835068786147744
```

Καλύτερη απόδοση έχει ο αλγόριθμος Random Forest.

Αρχεία για μοντέλα Neural Networks, Random Forest, Bayesian Networks: **2_neural_model.py**, **2_random_model.py**, **2_bayesian_model.py**

Αρχείο για μετρικές και γραφήματα: **2_metrics.py** (αλλαγή των αρχείων γραμμές 39-40 ανάλογα με το μοντέλο)

Χρόνοι εκτέλεσης μοντέλων: ~ 1h, για τον λόγο αυτό τα αρχεία αποτελεσμάτων των μοντέλων δίνονται για να μπορούν να χρησιμοποιηθούν στις μετρικές.

Ομαδοποίηση

Για να χωρίσουμε τις εγγραφές μας σε συστάδες ανάλογα με την δραστηριότητά τους χρησιμοποιήσαμε τους αλγορίθμους συσταδοποίησης kmeans και MiniBatchKMeans.

Στην αρχή διακριτοποιούμε τις τιμές των στηλών των δεδομένων μας σε bins. Επιπλέον, τις κανονικοποιούμε με την συνάρτηση StandardScaler, έτσι ώστε οι μεταβλητές να έχουν την ίδια κλίμακα.

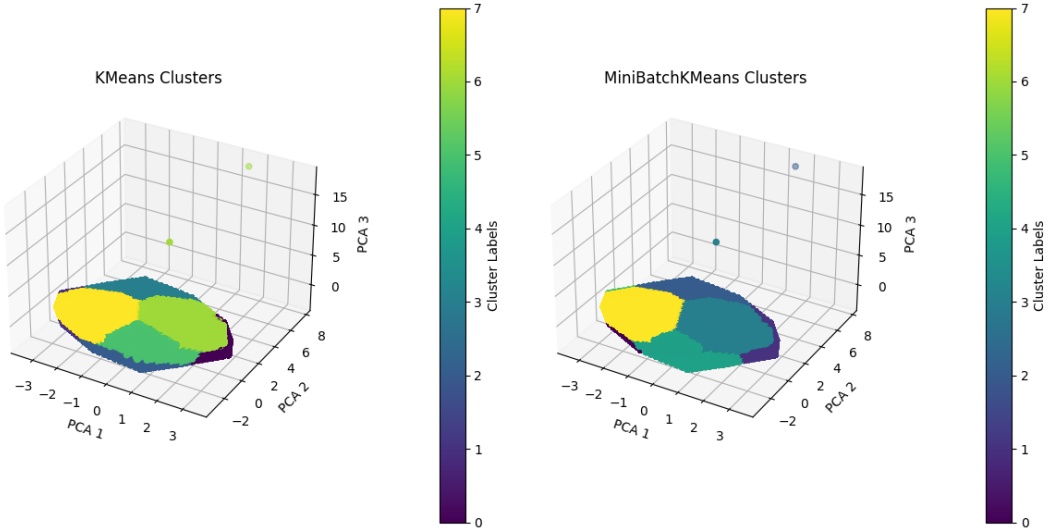
Έπειτα χρησιμοποιήθηκε ο αλγόριθμος PCA, έτσι ώστε να μειώσουμε τα δεδομένα σε 3 διαστάσεις, για να απεικονίσουμε καλύτερα τα δεδομένα.

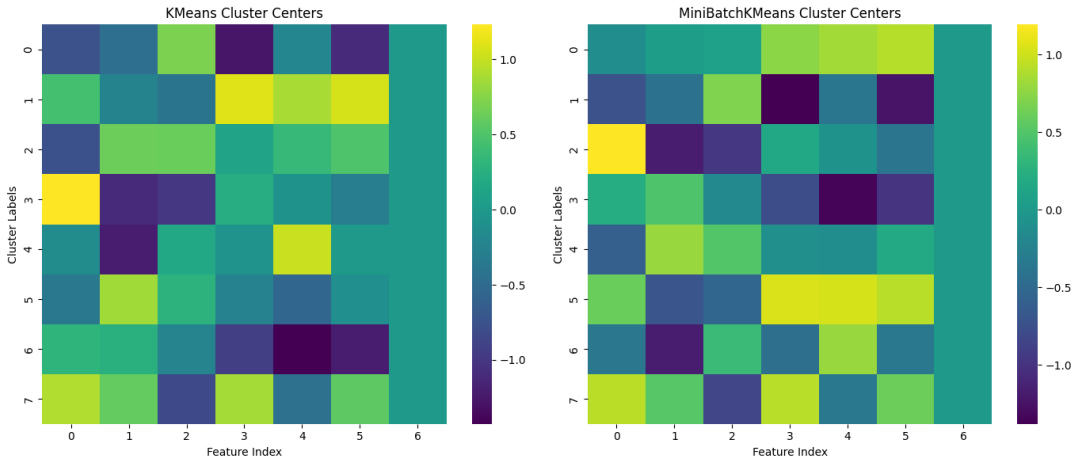
Στην συνέχεια χρησιμοποιούμε τους αλγόριθμους της sklearn KMeans και MiniBatchKMeans με αριθμό cluster 8. Ο αριθμός που επιλέχθηκε είναι μικρότερος από τον συνολικό αριθμό κλάσεων που είχαμε, έτσι ώστε να πετύχουμε μια πιο συγκεντρωτική ανάλυση δεδομένων. Επίσης, με μικρότερο αριθμό clusters θα προκύψουν πιο σαφείς ομάδες, γεγονός που θα βοηθήσει στην κατανόηση των μοτίβων.

Για την σύγκριση των δύο μοντέλων υπολογίστηκε το συνολικό άθροισμα των αποστάσεων των cluster, καθώς και ο μέσος όρος απόστασης από το κέντρο. Τα αποτελέσματα φαίνονται παρακάτω:

```
Sum of Distances for KMeans: 18900602.769582752
Sum of Distances for MiniBatchKMeans: 18852057.456348747
Average Distance to Cluster Center for KMeans: 2.925188563339108
Average Distance to Cluster Center for MiniBatchKMeans: 2.9176753534797717
```

Ο MiniBatchKMeans έχει μικρότερη τιμή συνολικού αθροίσματος αποστάσεων, που σημαίνει ότι τα clusters που παράγει είναι πιο συμπαγή. Παρόλα αυτά, δεν έχουν μεγάλη διαφορά μεταξύ τους. Η μέση απόσταση από τα κέντρα των clusters είναι επίσης λίγο μικρότερη στον MiniBatchKMeans. Συμπεραίνουμε ότι ο MiniBatchKMeans είναι λίγο καλύτερος από τον kmeans στο συγκεκριμένο dataset. Παρακάτω βλέπουμε τα scatterplots των δύο μεθόδων, καθώς και τους heatmaps. Τα γραφήματα scatter φαίνονται παρόμοια, με τις δύο μεθόδους να έχουν καταλήξει σε παρόμοια clusters. Τα γραφήματα heatmap παρέχουν μια πιο λεπτομερή ανάλυση για τα clusters. Καθώς ο αλγόριθμος kmeans είναι πιο ακριβής στην εύρεση των κέντρων, ενώ ο MiniBatch μέσω των επαναλήψεων γίνεται πιο γρήγορος, αλλά λιγότερο ακριβής, παρατηρούμε διαφορές στα κεντρικά σημεία των clusters.





Αρχείο για clustering: **3_clustering.py**