ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΜΕΘΟΔΟΙ ΕΞΟΡΥΞΗΣ ΓΝΩΣΗΣ ΑΠΟ ΔΕΔΟΜΕΝΑ ΥΓΕΙΑΣ ΣΥΝΕΧΟΥΣ ΡΟΗΣ

Βασιλική Χ. Κουτσοχρήστου

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Σεπτέμβριος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ ΚΑΙ ΑΣΦΑΛΙΣΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ

Μέθοδοι εξόρυξης γνώσης από δεδομένα υγείας συνεχούς ροής

Βασιλική Χ. Κουτσοχρήστου

Διατριβή

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην Εφαρμοσμένη Στατιστική

Πειραιάς
Σεπτέμβριος 2024

UNIVERSITY OF PIRAEUS

DEPARTMENT OF STATISTICS AND INSURANCE SCIENCE

Data Mining from medical data streams

By

Vasiliki C. Koutsochristou

Thesis

submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfilment of the requirements for the degree of Master of Science in Applied Statistics

Piraeus, Greece

September 2024

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμό. …….. συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών.

Τα μέλη της Επιτροπής ήταν:

- Καθηγητής Μ. Κούτρας (Επιβλέπων)
- Καθηγητής Δ. Κυριαζής
- Αναπληρωτής Καθηγητής Ν. Πελέκης

Η έγκριση της Διπλωματική Εργασίας από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

Στην αδερφή μου

Ελευθερία

Περίληψη

Με την ανάπτυξη των νέων τεχνολογιών, τα δεδομένα συνεχούς ροής εμφανίζονται ολοένα και συχνότερα σε πληθώρα εφαρμογών όπως η τεχνητή νοημοσύνη, η ανίχνευση απάτης, η επιχειρηματική ευφυΐα αλλά και η υγεία. Για το λόγο αυτό είναι απαραίτητη η εύρεση τεχνικών για την εξόρυξη γνώσης από αυτά. Ειδικότερα στον τομέα της υγείας, η κατάλληλη επεξεργασία των δεδομένων συνεχούς ροής μπορεί να αποβεί σωτήρια και να εξάγει σημαντικά συμπεράσματα. Η παρούσα διπλωματική διερευνά τις διάφορες μεθόδους προετοιμασίας των δεδομένων ανιχνεύοντας ελλιπή, ασυνεπή δεδομένα καθώς επίσης και ακραίες τιμές με τη χρήση κατάλληλων μεθόδων.

Παράλληλα αξιοποιούνται μέθοδοι ανάλυσης των δεδομένα. Πιο συγκεκριμένα διερευνόνται μεθόδοι σύνοψης, προκειμένου να επιλυθούν τα ζητήματα που προκύπτουν λόγω του τεράστιου όγκου των δεδομένων. Τέλος η διπλωματική ασχολείται με την συσταδοποίηση των δεδομένων συνεχούς ροής, αλλά και με την εύρεση συχνών μοτίβων.

Abstract

With the development of new technologies, streaming data is becoming increasingly common in a variety of applications such as artificial intelligence, fraud detection, business intelligence and health. For this reason it is necessary to find techniques to extract knowledge from them. In the health sector in particular, appropriate processing of continuous flow data can be life-saving and draw important conclusions. This thesis investigates the various methods of data preparation by detecting incomplete, inconsistent data as well as outliers using appropriate methods.

At the same time, this thesis analyses the data. More specifically, it explores methods of summarizing data in order to resolve the issues that arise due to the huge amount of data. Finally, it deals with the aggregation of continuous flow data, but also with finding frequent patterns.

Περιεχόμενα

Για την παρούσα διπλωματική εργασία χρησιμοποιήθηκε εξοπλισμός του Πανεπιστημίου Πειραιώς που αποκτήθηκε με χρηματοδότηση της πράξης «Προμήθεια και Εγκατάσταση Εξειδικευμένου Ερευνητικού Εξοπλισμού στο Πανεπιστήμιο Πειραιώς» (MIS 5066760) του Περιφερειακού Επιχειρησιακού Προγράμματος «Αττική 2014-2020»

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κατάλογος Συντομογραφιών

MCAR Missing Completely at Random
MAR Missing at Random
MNAR Missing not at Random
LOF Local Outlier Factor
LRD Local reachability Density
ECM Evolving Clustering Method
DBI Davies-Bouldin Index
DBIECM Density Based Interval Estimation Clustering

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Τι είναι τα δεδομένα συνεχούς ροής

Τα δεδομένα συνεχούς ροής αποτελούν μία συνεχή, διατεταγμένη στο χρόνο αλυσίδα από δεδομένα, που δεν έχει ούτε αρχή ούτε τέλος. Έχουν τεράστιο όγκο, ρέουν με μεγάλη ταχύτητα και έχουν άπειρο μήκος, ενώ η σειρά με την οποία έρχονται δεν δύναται να ελεγχθεί. Αν η πληροφορία δεν επεξεργαστεί και δεν αποθηκευτεί άμεσα, τότε χάνεται. Εύκολα γίνεται αντιληπτό πως διαφέρουν σημαντικά από τα στατικά, δυναμικά δεδομένα. Παρακάτω παρατίθενται οι βασικές διαφορές ανάμεσα στην εξόρυξη γνώσης από στατικά δεδομένα και από δεδομένα συνεχούς ροής.

Παράγοντες Σύγκρισης	**Στατικά Δεδομένα**	**Δεδομένα Συνεχούς ροής**
Μέγεθος	Συγκεκριμένο Μέγεθος	Άπειρο Μέγεθος
Προσβασιμότητα	Πολλαπλή προσβασιμότητα	Μόνο ένα πέρασμα
Διαδικασία Επεξεργασίας	Επεξεργασία όλων των δεδομένων σε παρτίδες	Επεξεργασία δείγματος (online)
Χρόνος επεξεργασίας	Άπειρος χρόνος	Περιορισμένος Χρόνος
Μνήμη	Άπειρη Μνήμη	Περιορισμένη Μνήμη
Ακρίβεια Αποτελέσματος	Ακρίβεια στα Αποτελέσματα	Προσεγγιστικά Αποτελέσματα

Πίνακας 1-1: Διαφορές στατιστικών δεδομένων και δεδομένων Συνεχούς ροής

Η επεξεργασία και η ανάλυση των δεδομένων συνεχούς ροής κρίνεται απαραίτητη για την εξόρυξη γνώσης, η οποία μπορεί να χρησιμοποιηθεί για να βελτιώσει πολλές διαδικασίες,

μεταξύ των οποίων, λήψη αποφάσεων, καθορισμό στόχων, μείωση κινδύνου απάτης και πρόβλεψη. Σε αντίθεση με τα στατικά δεδομένα που είναι αποθηκευμένα σε βάσεις, τα δεδομένα συνεχούς ροής έχουν κατανομή που αλλάζει στο χρόνο. Για το λόγο αυτό είναι σημαντικό, το μοντέλο που θα χρησιμοποιηθεί για την επεξεργασία τους, να προσαρμόζεται, να ενημερώνεται και να επανεκπαιδεύεται με την εξέλιξή τους. Όλα τα παραπάνω θέτουν σημαντικά εμπόδια στη συμβατική εξόρυξη και στη χρήση των τεχνικών μηχανικής μάθησης που αφορούν τα στατικά δεδομένα.

Κάθε στοιχείο της ροής είναι ένα ζεύγος $(\boldsymbol{X}_i, T_i)$. Με $\boldsymbol{X}_i$, (όπου i η i -οστή παρατήρηση της ροής με $i = 1, \ldots, N$ και $N \to \infty$), συμβολίζεται το διάνυσμα, στο οποίο περιέχονται τα δεδομένα, που φτάνει στο χρόνο T_i. Με T_i συμβολίζεται η χρονική σήμανση που καθορίζει πότε δημιουργήθηκε το $\boldsymbol{X}_i$. Τα δεδομένα συλλέγονται συνήθως από smartphones ή και άλλες ψηφιακές πηγές και προέρχονται από περιοχές όπως τα οικονομικά, οι διαδικτυακές εφαρμογές, η διαχείριση ασφάλειας δικτύου, η δημόσια υγεία, οι στρατιωτικές εφαρμογές. Μπορούν επίσης να παραχθούν από γεννήτριες ροών δεδομένων. Πρόκειται για γεννήτριες που παράγουν μια ροή από δεδομένα, προσομοιώνοντας ένα δυναμικό περιβάλλον όπου τα δεδομένα φτάνουν διαδοχικά. Αποτελούν μια φθηνή πηγή δεδομένων, καθώς δημιουργούνται κατόπιν ζήτησης και έτσι αποφεύγεται η αποθήκευσή τους. Υπάρχουν πολλές γεννήτριες ροών δεδομένων όπως η Random RBF Generator, η SEA Concepts Generator, Hyperplane Generator και η Random Tree Generator (Mansour, Abdullah, 2022).

1.2 Τι είναι τα δεδομένα υγείας συνεχούς ροής

Με την ανάπτυξη τεχνικών για την ανάλυση δεδομένων συνεχούς ροής, επήλθε σημαντική εξέλιξη και στον τομέα της υγείας. Η αναβάθμιση των ιατρικών οργάνων μέτρησης και τεχνολογιών, επέτρεψε σε ιατρικές πληροφορίες αυξημένης κλίμακας, που αποτελούνται μεταξύ άλλων από εικόνες και ζωτικούς δείκτες, να καταγράφονται με ακρίβεια. Πιο συγκεκριμένα, οι πάροχοι υγείας στρέφονται ολοένα και περισσότερο στην διαρκή επεξεργασία των δεδομένων αυτών, προκειμένου να παρακολουθήσουν την υγεία των ασθενών τους σε πραγματικό χρόνο.

Η συνεχής συλλογή, επεξεργασία και ανάλυση μεγάλων ποσοτήτων δεδομένων υγείας συνεχούς ροής συμβάλει καθοριστικά στην βελτίωση της υγείας, της περίθαλψης και της φροντίδας των ασθενών, καθώς στοχεύει στην αύξηση της ακρίβειας και της αποτελεσματικότητας των διαγνώσεων, όπως επίσης και στην πρόληψη. Πιο συγκεκριμένα βοηθά στην παρακολούθηση ζωτικών μετρήσεων, στην καταγραφή δοσολογιών φαρμάκων και σε άλλες σχετικές παραμέτρους, μειώνοντας έτσι των κίνδυνο εμφάνισης ανεπιθύμητων ασθενειών και επεισοδίων. Επιπλέον χρησιμοποιείται για τον προσδιορισμό μοτίβων που υποδηλώνουν την εμφάνιση ορισμένων επιπλοκών, βοηθώντας έτσι στην ανίχνευση και την ταχύτερη ανταπόκριση, ώστε να γίνονται νωρίτερα επεμβάσεις, όταν αυτό είναι απαραίτητο. Αποτελούν δηλαδή μέτρο πρόληψης.

Αξιοσημείωτη είναι η χρήση φορητών συσκευών όπως τα έξυπνα ρολόγια, που επιτρέπουν στους ασθενείς να παρακολουθούν οι ίδιοι τα δεδομένα υγείας τους. Συγχρόνως δίνεται η δυνατότητα και στους παρόχους υγείας να παρακολουθήσουν εξ αποστάσεως την υγεία των ασθενών τους και να λάβουν γρήγορες και στοχευμένες αποφάσεις.

Ωστόσο είναι σημαντικό να τονιστεί πως η διαχείριση των δεδομένων υγείας συνεχούς ροής εμφανίζει διάφορες προκλήσεις. Μία από αυτές είναι η διασφάλιση της ιδιωτικότητάς τους, όταν αυτές αφορούν σε πραγματικούς ασθενείς, της διαλειτουργικότητας και της ποιότητάς τους. Για την επιτυχή ανάλυση των δεδομένων είναι απαραίτητο να ξεπερασθούν όλες οι δυσκολίες και να αξιοποιηθούν σωστά και αποτελεσματικά οι τεχνολογίες ανάλυσης δεδομένων συνεχούς ροής (Shukla, 2023).

1.3 Σύνοψη δεδομένων συνεχούς ροής

Όπως αναφέρθηκε, λόγω του τεράστιου όγκου τους, τα δεδομένα συνεχούς ροής δε δύνανται να αποθηκευτούν στο σύνολό τους. Για το λόγο αυτό, στα πλαίσια της εξόρυξης, δημιουργείται η σύνοψη των δεδομένων ως μία περίληψη της πληροφορίας τους. Η υλοποίησή της είναι μία περιοδική διαδικασία, που επιλέγει ένα υποσύνολο της ροής των δεδομένων. Ένας τρόπος, είναι να επιλεχθεί ένα μέρος των δεδομένων τυχαία από το σύνολο. Μπορούν επίσης να

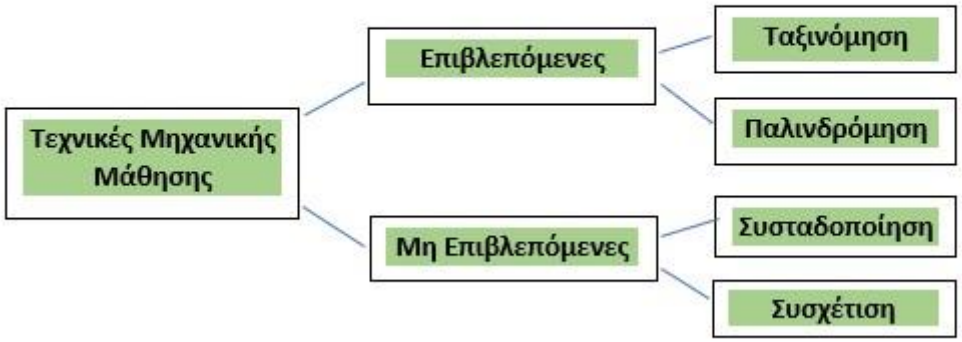
επιλεχθούν τα πιο πρόσφατα δεδομένα. Οι κύριες μέθοδοι σύνοψης των δεδομένων συνεχούς ροής είναι η δειγματοληψία, η λήψη δηλαδή δείγματος από το αρχικό σύνολο δεδομένων και το windowing. Με τον όρο windowing περιγράφεται η τμηματοποίηση της ροής δεδομένων σε μικρά πακέτα και στη συνέχεια η επιλογή ορισμένων εξ αυτών.

Είναι σημαντικό να τονιστεί πως, αν και η σύνοψη μειώνει το πλήθος των δεδομένων και κατ' επέκταση τον όγκο της πληροφορίας προς επεξεργασία και το κόστος ανάλυσης και αποθήκευσης, αυξάνει τα πιθανά σφάλματα και ενδεχομένως αλλοιώνει το αποτέλεσμα. Κρίνεται λοιπόν απαραίτητη η εύρεση μίας κατάλληλης μεθόδου για την κατασκευή σύνοψης, ώστε να μη χαθεί σημαντική πληροφορία κατά την επιλογή των δεδομένων (Mansour, Abdullah, 2022).

1.4 Τελική επεξεργασία δεδομένων συνεχούς ροής

Η επεξεργασία των δεδομένων συνεχούς ροής είναι μία συνεχής διαδικασία που εφαρμόζεται στα δεδομένα σε πραγματικό χρόνο, προκειμένου να δώσει άμεσα αποτελέσματα. Για την εξόρυξη γνώσης υπάρχουν πολλές προσεγγίσεις. Μία από αυτές είναι η μηχανική μάθηση. Στη μηχανική μάθηση οι εφαρμογές μαθαίνουν από τα δεδομένα και βελτιώνονται με την εμπειρία. Οι αλγόριθμοι εκπαιδεύονται για να βρουν μοτίβα και συσχετίσεις σε μεγάλα σύνολα δεδομένων και να λαμβάνουν τις καλύτερες αποφάσεις και προβλέψεις με βάση αυτή τους την ανάλυση. Οι εφαρμογές μηχανικής μάθησης βελτιώνονται με τη χρήση και γίνονται

πιο ακριβείς όσο περισσότερα δεδομένα έχουν πρόσβαση. Στο παρακάτω σχήμα παρουσιάζονται οι βασικές τεχνικές μηχανικής μάθησης.



Σχήμα 1-1:Τεχνικές Μηχανικής Μάθησης

Στην επιβλεπόμενη μάθηση, τα μοντέλα εκπαιδεύονται σε ένα επισημασμένο σύνολο δεδομένων, που χρησιμοποιείται για την εκπαίδευση αλγορίθμων ταξινόμησης δεδομένων και πρόβλεψης με μεγάλη ακρίβεια. Τα "επισημασμένα" δεδομένα αναφέρονται σε κάποια δεδομένα εισόδου που έχουν ήδη επισημανθεί με τη σωστή έξοδο. Δηλαδή, κάθε σημείο δεδομένων εισόδου, σχετίζεται με την αντίστοιχη ετικέτα (label) εξόδου. Χρησιμοποιείται σε εφαρμογές για αξιολόγηση κινδύνου, ανίχνευση απάτης, φιλτράρισμα ανεπιθύμητων μηνυμάτων και πολλά ακόμη. Στην επιβλεπόμενη μάθηση, τα δεδομένα με ετικέτα είναι δεδομένα που περιέχουν τα χαρακτηριστικά (μεταβλητές X) και τον στόχο (μεταβλητή Y).

Στην μη επιβλεπόμενη μάθηση, χρησιμοποιούνται δεδομένα χωρίς ετικέτα. Από αυτά τα δεδομένα, ανακαλύπτονται μοτίβα και ομοιότητες που βοηθούν στην επίλυση ζητημάτων Συσταδοποίησης (Clustering) και Συσχέτισης (Association).

- **Ταξινόμηση**

Η ταξινόμηση (Classification) είναι μια επιβλεπόμενη μέθοδος μηχανικής μάθησης όπου το μοντέλο προσπαθεί να κατηγοριοποιήσει τα δεδομένων σε προκαθορισμένες κλάσεις ή κατηγορίες με βάση τα χαρακτηριστικά τους. Στην ταξινόμηση, το μοντέλο εκπαιδεύεται πλήρως χρησιμοποιώντας τα δεδομένα εκπαίδευσης και στη συνέχεια αξιολογείται σε δεδομένα δοκιμών πριν χρησιμοποιηθεί για την εκτέλεση πρόβλεψης σε νέα, άγνωστα δεδομένα. Ο

κύριος στόχος της ταξινόμησης είναι η κατασκευή ενός μοντέλου που μπορεί να αντιστοιχίσει με ακρίβεια μια ετικέτα ή μια κατηγορία σε μια νέα παρατήρηση με βάση τα χαρακτηριστικά της. Για παράδειγμα, ένα μοντέλο ταξινόμησης μπορεί να εκπαιδευτεί σε ένα σύνολο δεδομένων εικόνων με χειρόγραφους αριθμούς από το ένα έως το δέκα. Στη συνέχεια θα χρησιμοποιηθεί για την πρόβλεψη της κατηγορίας νέων εικόνων αριθμών με βάση τα χαρακτηριστικά τους όπως το σχήμα, οι γωνίες, η κλίση.

- **Παλινδρόμηση**

Η παλινδρόμηση (Regression) είναι μια επιβλεπόμενη διαδικασία μηχανικής μάθησης που χρησιμοποιείται για την ανάλυση της σχέσης μεταξύ μιας εξαρτημένης μεταβλητής και μιας ή περισσότερων ανεξάρτητων μεταβλητών. Στόχος είναι να προσδιοριστεί η καταλληλότερη συνάρτηση που χαρακτηρίζει τη σύνδεση μεταξύ αυτών των μεταβλητών και να βρεθεί το μοντέλο που μπορεί να χρησιμοποιηθεί για να γίνουν προβλέψεις ή να εξαχθούν συμπεράσματα. Αφορά δηλαδή στη δημιουργία μοντέλων πρόβλεψης αριθμητικών τιμών και συγκεκριμένα της τιμής της εξαρτημένης μεταβλητής. Μπορούν να χρησιμοποιηθούν πολλά διαφορετικά μοντέλα. Το πιο απλό είναι η γραμμική παλινδρόμηση που προσπαθεί να προσαρμόσει δεδομένα στο καλύτερο υπερ-επίπεδο που διέρχεται από αυτά.

- **Συσταδοποίηση**

Με τον όρο Συσταδοποίηση (Clustering), αναφερόμαστε σε μη επιβλεπόμενη διαδικασία μηχανικής μάθησης που σκοπό έχει τον προσδιορισμό συστάδων – κατηγοριών που περιγράφουν κατάλληλα ένα σύνολο δεδομένων, έτσι ώστε τα αντικείμενα να είναι όμοια (ή συσχετιζόμενα) σε κάθε συστάδα και διαφορετικά (ή μη συσχετιζόμενα) από τα αντικείμενα των άλλων συστάδων. Αξίζει να σημειωθεί πως σε αντίθεση με την κατηγοριοποίηση, στη συσταδοποίηση, οι κατηγορίες που δημιουργούνται δεν είναι από την αρχή γνωστές.

- **Συσχέτιση**

Η συσχέτιση (Association) είναι μια μέθοδος μη επιβλεπόμενης μηχανικής μάθησης για την ανακάλυψη μοτίβων και σχέσεων μεταξύ μεταβλητών σε μεγάλες βάσεις δεδομένων. Χρησιμεύει στην περιγραφή της δομής των δεδομένων και στην εξερεύνηση των

χαρακτηριστικών τους, βοηθώντας άλλες διαδικασίες όπως η Ταξινόμηση και η Συσταδοποίηση. Είναι ευρέως διαδεδομένη στην ανάλυση του καλαθιού της αγοράς, όπου αναζητούνται κανόνες και σχέσεις μεταξύ των αντικειμένων που αγοράζονται μαζί. Ένας τυπικός κανόνας συσχέτισης σε μια ανάλυση καλαθιού αγοράς μπορεί να αναφέρει ότι εάν ένας πελάτης αγοράσει ψωμί και βούτυρο (X), είναι πιθανό να αγοράσει και γάλα (Y). Σκοπός είναι να προσδιοριστούν όλοι οι κανόνες συσχέτισης της μορφής $X \rightarrow Y$, όπου τα X και Y είναι ασύνδετα στοιχεία $(X \cap Y = \emptyset)$, οι οποίοι ξεπερνούν τα κατώφλια ελάχιστης υποστήριξης και ελάχιστης εμπιστοσύνης. Με τον όρο υποστήριξη (support), αναφέρεται το ποσοστό των δοσοληψιών που περιέχουν το $X \cup Y$ ή αλλιώς η πιθανότητα $P(X \cup Y)$. Με τον όρο εμπιστοσύνη (confidence), αναφέρεται η δεσμευμένη πιθανότητα $P(Y|X) = \frac{P(X \cup Y)}{P(X)}$.

1.5 Προβλήματα και περιορισμοί

Στις περισσότερες τεχνικές εξόρυξης, η ποσότητα των δεδομένων είναι περιορισμένη, οπότε μπορεί να αποθηκευτεί και να αναλυθεί εύκολα από έναν αλγόριθμο. Για τα δεδομένα συνεχούς ροής ωστόσο υπάρχουν περιορισμοί, που δυσκολεύουν τόσο τη συλλογή, όσο και την επεξεργασία τους.

Αρχικά, νέα δεδομένα καταφθάνουν συνεχώς, τα οποία πρέπει να επεξεργαστούν άμεσα ειδάλλως θα χαθούν. Επιπλέον δεν υπάρχει έλεγχος στη σειρά με την οποία θα πρέπει να γίνει η επεξεργασία τους, καθώς αυτά φτάνουν με τυχαίο τρόπο. Επιπρόσθετα το μέγεθος της ροής είναι άπειρο. Συνεπώς δεν είναι εφικτή η αποθήκευσή τους λόγω περιορισμένου χώρου. Στην πράξη, αποθηκεύεται μέρος της πληροφορίας για συγκεκριμένο χρόνο, το οποίο στη συνέχεια διαγράφεται. Χρησιμοποιείται δηλαδή από τον αλγόριθμο ένας μηχανισμός λήθης. Τέλος η κατανομή από την οποία παράγονται τα δεδομένα δεν είναι σταθερή. Αλλάζει με την πάροδο του χρόνου και για αυτό ο αλγόριθμός επεξεργασίας πρέπει να είναι ευέλικτος και να προσαρμόζεται άμεσα.

Για όλους τους παραπάνω λόγους, οι συμβατικές μέθοδοι εξόρυξης δεδομένων δεν έχουν εφαρμογή στα δεδομένα συνεχούς ροής. Κρίνεται λοιπόν απαραίτητη η ανάπτυξη νέων

αποτελεσματικών αλγορίθμων για την καταλληλότερη και αποτελεσματικότερη επεξεργασία και ανάλυσή τους.

1.6 Δομή της διπλωματικής εργασίας

Για την ανάλυση στην παρούσα διπλωματική, θα χρησιμοποιηθούν δεδομένα τα οποία αφορούν σε ασθενείς με covid που παρακολουθήθηκαν με έξυπνα ρολόγια. Θα πρέπει να σημειωθεί ότι στα δεδομένα αυτά έχουν εφαρμοστεί τεχνικές ανωνυμοποίησης, ώστε να μην μπορούν με κανένα τρόπο να συσχετιστούν με πραγματικούς ανθρώπους.

Αρχικά τα δεδομένα θα προετοιμαστούν για την επεξεργασία τους. Θα ανιχνευτούν ελλιπή, ασυνεπή δεδομένα αλλά και ακραίες τιμές μέσω της οπτικοποίησης, του αλγορίθμου Local Outlier Factor και με τη χρήση της μέσης τιμής και της διασποράς. Στη συνέχεια για την ανάλυσή τους θα χρησιμοποιηθούν μέθοδοι σύνοψης και συγκεκριμένα η Δειγματοληψία το Windowing και τα Ιστογράμματα. Τέλος θα εφαρμοστούν συσταδοποίηση και κανόνες συσχετίσεων.

ΚΕΦΑΛΑΙΟ 2

Προετοιμασία των Δεδομένων

2.1 Προ επεξεργασία Δεδομένων

Η προεπεξεργασία δεδομένων είναι ένα σημαντικό βήμα στη διαδικασία ανάλυσης δεδομένων. Περιλαμβάνει τις ενέργειες που εφαρμόζονται στα δεδομένα για καθαρισμό και μορφοποίηση ώστε να είναι έτοιμα για ανάλυση. Δίνει επίσης μια πρώτη εικόνα των δεδομένων.

2.1.1 Προετοιμασία Δεδομένων

Η προετοιμασία των δεδομένων, προκειμένου αυτά να φτάσουν σε μία καταλληλότερη μορφή, και να εξάγουν ακριβή αποτελέσματα αποτελεί το μεγαλύτερο και πιο χρονοβόρο μέρος της εξόρυξης γνώσης. Τα δεδομένα στον πραγματικό κόσμο είναι "βρώμικα". Πιο συγκεκριμένα είναι:

1. **Ελλιπή:** μπορεί να λείπουν κάποιες τιμές γνωρισμάτων.
2. **Με θόρυβο**: περιέχουν λάθη ή ακραίες τιμές (outliers)
3. **Ασυνεπή:** περιέχουν μη λογικές συσχετίσεις ή διπλότυπες εγγραφές.

Οι τρεις αυτές κατηγορίες εξετάζονται στη συνέχεια.

Τα δεδομένα που λείπουν, προκύπτουν όταν δεν έχουν αποθηκευτεί δεδομένα για ορισμένες μεταβλητές. Μπορεί να χαθούν λόγω ελλιπούς εισαγωγής δεδομένων, δυσλειτουργιών εξοπλισμού, χαμένων αρχείων, αλλά και πολλών άλλων λόγων. Είναι σημαντικό να ληφθεί υπόψη ο λόγος ύπαρξης ελλιπών δεδομένων, καθώς βοηθά να προσδιοριστεί ο τύπος τους. Υπάρχουν τρεις κύριοι τύποι ελλιπών δεδομένων.

- **Εντελώς τυχαία ελλιπή δεδομένα**

Ο λόγος ύπαρξης εντελώς τυχαίων ελλιπών δεδομένων (Missing Completely at Random - MCAR) δεν σχετίζεται με τα ίδια τα δεδομένα. Πρόκειται δηλαδή για δεδομένα που δεν καταγράφονται, για τυχαίους λόγους και τα οποία δεν σχετίζονται με τη συγκεκριμένη τιμή,

αλλά ούτε και με άλλες μεταβλητές. Για το λόγο αυτό η ανάλυση τους είναι αμερόληπτη, καθώς η τυχαιότητα αυτή δεν οφείλεται σε κανένα παράγοντα. Για παράδειγμα, σε ένα σύνολο δεδομένων από ένα ερωτηματολόγιο, λείπουν ορισμένες απαντήσεις, χωρίς όμως οι απαντήσεις αυτές να ανήκουν σε κάποια συγκεκριμένη κατανομή, αλλά να έχουν ένα μεγάλο εύρος τιμών. Αυτό πιθανόν να οφείλεται σε άτομα τα οποία εγκατέλειψαν την έρευνα ή προσπέρασαν την ερώτηση.

Είναι σημαντικό να τονιστεί πως σπάνια τα ελλιπή δεδομένα είναι MCAR, καθώς η αληθινή τυχαιότητα είναι ασυνήθιστη. Εντελώς τυχαία ελλιπή δεδομένα, θεωρούνται επίσης τα δεδομένα που λείπουν λόγω κάποιου τεχνικού προβλήματος ή απώλειας δείγματος.

- **Τυχαία ελλιπή δεδομένα**

Τα τυχαία ελλιπή δεδομένα (Missing at Random - MAR), στην πραγματικότητα δε λείπουν εντελώς τυχαία. Ο λόγος απώλειας τους σχετίζεται με μία άλλη παρατηρούμενη μεταβλητή, αλλά όχι με τη συγκεκριμένη τιμή του ίδιου του δεδομένου. Για παράδειγμα σε ένα ερωτηματολόγιο παρατηρείται από μια συγκεκριμένη ηλικιακή ομάδα να μην απαντάει σε κάποιες ερωτήσεις. Οι απαντήσεις που παραλείπονται δεν είναι απαραίτητο να είναι ίδιες, τα δεδομένα δηλαδή δεν λείπουν λόγω του περιεχομένου τους. Η απώλεια τους οφείλεται στους ερωτηθέντες.

- **Μη τυχαία ελλιπή δεδομένα**

Τα δεδομένα είναι μη τυχαία ελλιπή (Missing not at Random – MNAR), αν η πιθανότητα απώλειας εξαρτάται από τα ίδια τα ελλιπή δεδομένα. Για παράδειγμα σε μία ερώτηση σχετικά με το εισόδημα, άτομα με χαμηλό μισθό είναι πιο δύσκολο να το αποκαλύψουν και για το λόγο αυτό στα τελικά δεδομένα θα υπάρχουν ελάχιστες χαμηλές τιμές. Τα μη τυχαία ελλιπή δεδομένα απαιτούν ειδική μεταχείριση και είναι αρκετά περίπλοκα στην ανάλυσή τους.

Συνοψίζοντας τα ελλιπή δεδομένα είναι προβληματικά, καθώς ανάλογα με τον τύπο τους μπορεί να προκαλέσουν προβλήματα μεροληψίας στη δειγματοληψία. Στην πράξη τα ελλιπή δεδομένα τύπου MCAR και MAR μπορούν να αγνοηθούν, καθώς δεν διαφέρουν σημαντικά από τις παρατηρούμενες τιμές. Τα ελλιπή δεδομένα τύπου MNAR ωστόσο διαφέρουν

σημαντικά από τις παρατηρούμενες τιμές και για το λόγο αυτό δεν μπορούν να αγνοηθούν, καθώς υπάρχει περίπτωση το δείγμα να μην αντικατοπτρίζει όλο το εύρος τιμών. Μπορεί δηλαδή να απουσιάζουν οι πολύ μικρές ή οι πολύ μεγάλες τιμές.

Στη συνέχεια εξετάζεται η κατηγορία που αφορά τα δεδομένα με θόρυβο. Τα outliers είναι ακραίες τιμές που διαφέρουν από τα περισσότερα δεδομένα του συνόλου. Έχουν μεγάλο αντίκτυπο στη στατιστική ανάλυση καθώς αλλοιώνουν τα αποτελέσματα. Η ανίχνευσή τους, όπως και η σωστή διαχείρισή τους, είναι σημαντική, ώστε τα συμπεράσματα που εξάγονται να είναι έγκυρα. Αξίζει να σημειωθεί πως ορισμένες ακραίες τιμές αντιπροσωπεύουν πραγματικές τιμές από τη φυσική διακύμανση του πληθυσμού και δεν προκύπτουν όλες από εσφαλμένη εισαγωγή δεδομένων, δυσλειτουργία εξοπλισμού ή άλλα σφάλματα μέτρησης. Για το λόγο αυτό συνίσταται προσοχή κατά τον καθαρισμό των δεδομένων. Οι πραγματικές ακραίες τιμές θα πρέπει να διατηρούνται, επειδή αντιπροσωπεύουν απλώς φυσικές παραλλαγές στο δείγμα.

Τέλος τα ασυνεπή δεδομένα εμφανίζονται σε περιπτώσεις που υπάρχουν τα ίδια δεδομένα σε διαφορετικές μορφές. Υπάρχουν δηλαδή διαφορετικά αρχεία που περιέχουν διαφορετικές πληροφορίες σχετικά με ένα συγκεκριμένοι άτομο ή αντικείμενο, με αποτέλεσμα να προκύπτουν αναξιόπιστες πληροφορίες

Η διόρθωση των τριών αυτών κατηγοριών “βρώμικων” δεδομένων, αποτελεί τη διαδικασία καθαρισμού που επιβάλλεται να ακολουθηθεί, προκειμένου να αυξηθεί η ποιότητα των δεδομένων και κατ’ επέκταση η αποτελεσματικότητα κατά την επεξεργασία τους (Heymans, Twisk, 2022).

2.1.2 Οπτικοποίηση

Η οπτικοποίηση των δεδομένων συνεχούς ροής αναφέρεται στην αναπαράσταση τους, καθώς αυτά παράγονται. Οι ροές δεδομένων απεικονίζονται δυναμικά, σε πραγματικό χρόνο, μέσω διαδραστικών γραφημάτων που ενημερώνονται καθώς φτάνουν νέα δεδομένα. Έτσι, δίνεται η δυνατότητα απόκτησης πληροφορίας και λήψης γρήγορων αποφάσεων, μιας και με την οπτική αναπαράσταση διακρίνονται με μεγαλύτερη ευκολία τάσεις, μοτίβα, αλλά και

ανωμαλίες. Παράλληλα δίνεται η δυνατότητα ταυτόχρονης οπτικοποίησης δεδομένων και σύγκρισης πολλαπλών συνεχών ροών δεδομένων, ενώ οι χρήστες μπορούν να αλληλοεπιδράσουν με τα γραφήματα κάνοντας ζουμ, φιλτράρισμα και επισήμανση ώστε να εστιάσουν σε σχετικές πληροφορίες.

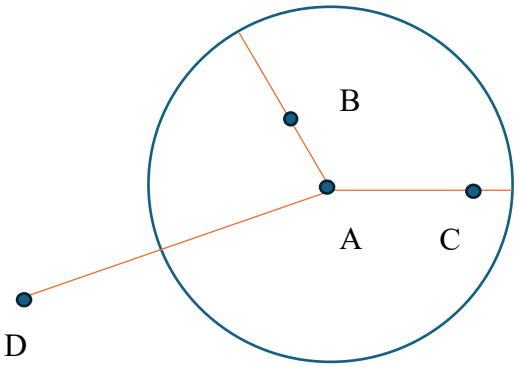
Η οπτικοποίηση σε πραγματικό χρόνο είναι ένα κρίσιμο εργαλείο για διάφορους κλάδους, καθώς επιτρέπει την άμεση κατανόηση και δράση με βάση τα δεδομένα ροής. Μέσω της προβολής των δεδομένων, εντοπίζονται άμεσα ανωμαλίες και τάσεις, τη στιγμή που αυτές συμβαίνουν, κάτι που είναι απαραίτητο σε περιπτώσεις που ακόμα και μια μικρή καθυστέρηση προκαλεί προβλήματα, όπως στους καρδιακούς παλμούς ενός ασθενή. Η απεικόνιση βοηθά στην κατανόηση περίπλοκων αλληλεπιδράσεων σε μεμονωμένες εγγραφές, ενώ συγχρόνως διατηρείται μια επισκόπηση των συνεχώς εξελισσόμενων καταστάσεων, ώστε να μη χάνονται κρίσιμες πληροφορίες. Τα πολύπλοκα δεδομένα απλοποιούνται και γίνονται κατανοητά από όλους. Αξιοσημείωτη είναι και η γρήγορη προσαρμογή των γραφημάτων με την ενσωμάτωση νέων δεδομένων, καθώς αυτά προσαρμόζονται στη νέα πληροφορία και ανανεώνονται από μόνα τους. Παράλληλα η οπτικοποίηση, μπορεί να συνδυαστεί με αλγόριθμους μηχανικής μάθησης, πρόγνωσης και χρονοσειρές, παρέχοντας εικόνα για πιθανές μελλοντικές τάσεις. Αυτό είναι απαραίτητο για τη διαχείριση κινδύνων καθώς δίνεται η δυνατότητα προετοιμασίας για την έγκαιρη αντιμετώπισή τους.

Υπάρχουν πολλά εργαλεία για την υλοποίηση της οπτικοποίησης των δεδομένων. Χαρακτηριστικό παράδειγμα αποτελεί το Power Bi και το Tableau, που επιτρέπουν την διαδραστική δημιουργία αναφορών και γραφημάτων σε διαφορετικές μορφές. Αν και χρησιμοποιούνται κυρίως από εταιρείες για σκοπούς επιχειρηματικής ευφυίας, το περιβάλλον και ο σχεδιασμός τους είναι φιλικά προς τον χρήστη, με αποτέλεσμα να μπορούν να αξιοποιηθούν και από λιγότερο έμπειρα άτομα. Παράλληλα υπάρχει πληθώρα σεμιναρίων για την εκμάθηση και της εξοικείωση με αυτά.

2.1.4 Local Outlier Factor

Πρόκειται για έναν αλγόριθμο που χρησιμοποιείται για μη επιβλεπόμενη ανίχνευση ακραίων τιμών. Μετρώντας την απόσταση του κάθε σημείου από τα γειτονικά του σημεία και συγκρίνοντάς την με την αντίστοιχη απόσταση των υπόλοιπων μη γειτονικών σημείων, δημιουργεί ένα σκόρ. Η απόσταση αυτή, ονομάζεται απόσταση προσβασιμότητας. Ειδικότερα, ο αλγόριθμος βασίζεται σε μια έννοια τοπικής πυκνότητας, χρησιμοποιώντας την απόσταση από τους k πλησιέστερους γείτονες, για την εκτίμηση της. Συγκρίνοντας την τοπική πυκνότητα ενός αντικειμένου με τις τοπικές πυκνότητες των γειτόνων του, εντοπίζονται περιοχές παρόμοιας πυκνότητας. Παράλληλα παρατηρούνται και σημεία που έχουν σημαντικά χαμηλότερη πυκνότητα από τους γείτονές τους. Τα σημεία αυτά θεωρούνται ακραίες τιμές.

Δημιουργείται έτσι για κάθε δεδομένο σημείο ένας νοητός κύκλος με κέντρο το σημείο αυτό και την απαιτούμενη ακτίνα ώστε να περιλαμβάνονται μέσα σε αυτόν οι k πλησιέστεροι γείτονές του. Αξίζει να σημειωθεί, ότι ένα σημείο μπορεί να έχει περισσότερους από k γείτονες, σε περίπτωση που υπάρχουν περισσότερα από ένα σημεία τα οποία ισαπέχουν από αυτό. Για να μειωθούν οι διακυμάνσεις, ο αλγόριθμος Local Outlier Factor θεωρεί ότι το σημείο απέχει από τους k γείτονες απόσταση ίση με την ακτίνα του κύκλου. Για κάθε εξωτερικό σημείο ωστόσο, η απόσταση ισούται με την πραγματική απόσταση των δύο σημείων. Το παραπάνω γίνεται εύκολα αντιληπτό στο σχήμα που ακολουθεί. Τα σημεία B και C απέχουν απόσταση μικρότερη από την ακτίνα του νοητού κύκλου. Ωστόσο στον Local Outlier Factor η απόσταση αυτή εξισώνεται με την ακτίνα του νοητού κύκλου.



Σχήμα 2-1:Απόσταση σημείων στον αλγόριθμο Local Outlier Factor

Οι αποστάσεις προσβασιμότητας από τους k-πλησιέστερους γείτονες ενός σημείου υπολογίζονται για τον προσδιορισμό της πυκνότητας τοπικής προσβασιμότητας (Local Reachability Densit - LRD) του σημείου αυτού. Πρόκειται για ένα μέτρο που ερμηνεύει την πυκνότητα των k-πλησιέστερων σημείων γύρω από ένα σημείο και υπολογίζεται ως το αντίστροφο της μέσης απόστασης προσβασιμότητας από τα k-πλησιέστερα γειτονικά αυτά σημεία. Διαισθητικά σύμφωνα με τον τύπο LRD, όσο μεγαλύτερη είναι η μέση απόσταση προσβασιμότητας (δηλαδή, οι γείτονες απέχουν πολύ από το σημείο), τόσο λιγότερη είναι η πυκνότητα των γύρω σημείων. Αυτό δηλώνει πόσο μακριά είναι ένα σημείο από το πλησιέστερο σύμπλεγμα σημείων. Οι χαμηλές τιμές του LRD υποδηλώνουν ότι το πλησιέστερο σύμπλεγμα απέχει πολύ από το σημείο. Όσο πιο κοντά είναι τα σημεία, η απόσταση είναι μικρότερη και η πυκνότητα είναι μεγαλύτερη. Παρακάτω δίνεται ο τύπος για τον υπολογισμό της πυκνότητας τοπικής προσβασιμότητας:

$$LRD_k(A) = [\frac{\sum_{B \in N_k(A)}\big(reachability - distance_k(A,B)\big)}{|N_k(A)|}]^{-1}$$

Το LRD κάθε σημείου χρησιμοποιείται για σύγκριση με το μέσο LRD των k γειτόνων του. Ο υπολογισμός του Local Outlier Factor (LOF) γίνεται λαμβάνοντας τον λόγο του μέσου όρου της πυκνότητας τοπικής προσβασιμότητας των k γειτόνων ενός σημείου και του της πυκνότητας τοπικής προσβασιμότητας αυτού του σημείου. Μια τιμή περίπου 1 υποδεικνύει ότι το αντικείμενο είναι συγκρίσιμο με τους γείτονές του (και επομένως όχι ακραίο). Μια τιμή κάτω από το 1 υποδηλώνει μια πιο πυκνή περιοχή , ενώ οι τιμές σημαντικά μεγαλύτερες από το 1 υποδηλώνουν ακραίες τιμές. Η εξίσωση για τον LOF φαίνεται παρακάτω:

$$LOF_k(A) = \frac{\sum_{B \in N_k(A)}(LRD_k(B))}{|N_k(A)|LRD_k(A)}$$

Ο Local Outlier Factor ανιχνεύει ευκολότερα τις ακραίες τιμές όταν η πυκνότητα των δεδομένων δεν είναι ίδια σε όλο το σύνολο δεδομένων και παρέχει καλύτερα αποτελέσματα από άλλες μεθόδους. Χαρακτηριστικά παραδείγματα εφαρμογής αποτελούν τα γεωγραφικά δεδομένα και οι ροές βίντεο. Ωστόσο, καθώς είναι αναλογία μεγεθών, είναι δύσκολο να ερμηνευτεί, διότι δεν υπάρχει οριακή τιμή πάνω από την οποία ένα σημείο ορίζεται ως ακραία τιμή. Ο προσδιορισμός της τιμής αυτής εξαρτάται από το εκάστοτε πρόβλημα και τον χρήστη (Pokrajac , Lazarevic, Latecki, 2007).

2.2 Σύνοψη δεδομένων

Με τη ραγδαία αύξηση της πληροφορίας απαιτείται ανάπτυξη αλγορίθμων, που θα μπορούν να επεξεργάζονται αποτελεσματικά σύνολα δεδομένων με τεράστιο όγκο. Για να επιτευχθεί αυτό, θα πρέπει η πληροφορία να συμπυκνωθεί. Ο όρος σύνοψη αναφέρεται στη δημιουργία ενός υποσυνόλου δεδομένων του αρχικού συνόλου. Η επιλογή των στοιχείων που το απαρτίζουν, θα πρέπει να γίνεται με τρόπο ώστε να διατηρείται όσο το δυνατόν μεγαλύτερο μέρος της πληροφορίας, χωρίς ωστόσο να συμπεριλαμβάνονται πολλά δεδομένα. Σκοπός λοιπόν είναι η εύρεση της σύνοψης που μεγιστοποιεί την ακρίβεια και την αποτελεσματικότητα. Η σύνοψη μπορεί να κατασκευαστεί με πολλούς τρόπους: τυχαία, χρησιμοποιώντας το μέσο όρο, με τα πιο πρόσφατα δεδομένα κ.ο.κ. Οι κύριες μέθοδοι κατασκευής σύνοψης είναι η δειγματοληψία, το windowing και τα ιστογράμματα.

2.2.1 Δειγματοληψία

Η δειγματοληψία (Sampling), είναι μία από τις απλούστερες μεθόδους για την κατασκευή σύνοψης σε δεδομένα συνεχούς ροής. Πραγματοποιείται επιλέγοντας ένα αντιπροσωπευτικό υποσύνολο δεδομένων, αρκετά μικρότερο από το αρχικό σύνολο. Η επιλογή γίνεται με τέτοιον τρόπον, ώστε τα εναπομείναντα δεδομένα να διατηρούν πολλά σημαντικά χαρακτηριστικά της ροής δεδομένων και μέσω αυτών εκτιμάται ένα σύνολο στατιστικών μέτρων , όπως ο μέσος

όρος, η διακύμανση και η κατανομή πιθανότητας. Σε αντίθεση με τη δειγματοληψία από ένα αποθηκευμένο σύνολο δεδομένων, η δειγματοληψία δεδομένων συνεχούς ροής εκτελείται παράλληλα την διέλευση των δεδομένων. Οποιοδήποτε στοιχείο δεν αποθηκεύεται στο δείγμα, χάνεται.

Υπάρχουν πολλές μέθοδοι δειγματοληψίας και η πρόκληση είναι να επιλεχθεί εκείνη η οποία θα παράγει ένα δείγμα που αντικατοπτρίζει τα στατιστικά στοιχεία των αρχικών δεδομένων. Η τυχαία δειγματοληψία είναι η απλούστερη μορφή δειγματοληψίας, κατά την οποία όλα τα δεδομένα έχουν ίσες πιθανότητες να συμπεριληφθούν στο δείγμα. Στα δεδομένα συνεχούς ροής ιδιαίτερα χρήσιμες είναι η δειγματοληψία σταθερού ποσοστού στη ροή και η διατήρηση ενός τυχαίου δείγματος σταθερού μεγέθους, οι οποίες και αναλύονται παρακάτω:

- **Δειγματοληψία σταθερού ποσοστού στη ροή**

Στην περίπτωση της δειγματοληψίας σταθερού ποσοστού, επιλέγεται ένα δείγμα από έναν πληθυσμό, με σταθερό ποσοστό, για παράδειγμα το 10%. Κάθε φορά δηλαδή αποθηκεύεται το 10% της συνολικής εισερχόμενης πληροφορίας. Η επιλογή πρέπει να γίνει προσεκτικά, ειδάλλως το δείγμα δεν θα είναι αντιπροσωπευτικό. Χαρακτηριστικό παράδειγμα αποτελεί η δειγματοληψία της ροής δεδομένων για εξόρυξη γνώσης σχετικά με τη διερεύνηση της συχνότητας με την οποία ένας χρήστης αναζητά το ίδιο ερώτημα δύο φορές. Αν παρθεί δείγμα από τις αναζητήσεις, θα προκύψουν εσφαλμένα αποτελέσματα, καθώς δεν θα συμπεριληφθούν όλες οι διπλές αναζητήσεις. Για το λόγο αυτό η δειγματοληψία πρέπει να γίνει στους χρήστες.

- **Διατήρηση ενός τυχαίου δείγματος σταθερού μεγέθους**

Για να διατηρηθεί σταθερό το μέγεθος ενός τυχαίου δείγματος, θα πρέπει στο τέλος της δειγματοληψίας να έχει ληφθεί συγκεκριμένος αριθμός δεδομένων. Εφόσον το μέγεθος του δείγματος είναι γνωστό, έστω n και απαιτείται μέγεθος δείγματος s, τότε κάθε δεδομένο του συνόλου έχει s/n πιθανότητα να επιλεγεί.

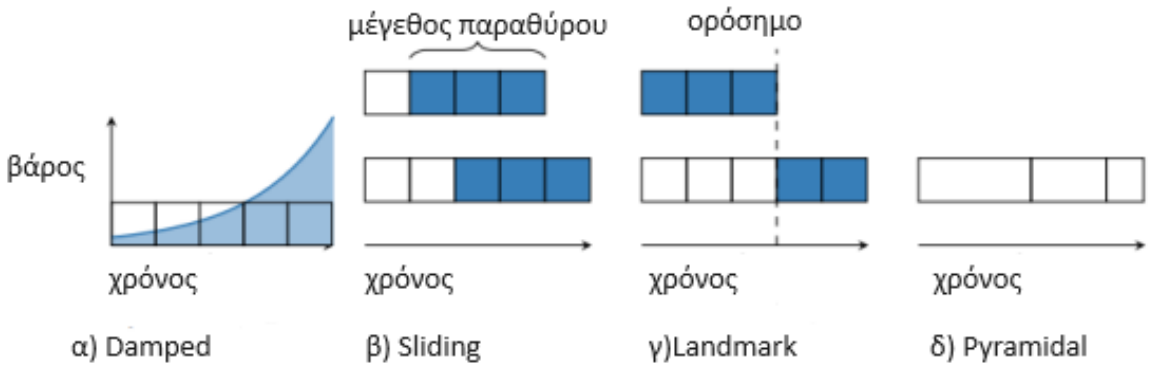
Το μέγεθος του δείγματος ωστόσο δεν είναι πάντα γνωστό. Σε αυτή την περίπτωση αν τη συγκεκριμένη χρονική στιγμή διατίθενται n δεδομένα, επιλέγονται s από αυτά. Κάθε επόμενο δεδομένο επιλέγεται με πιθανότητα s/k, όπου k η σειρά εμφάνισής του. Αν ένα δεδομένο

επιλεγεί και στο δείγμα υπάρχουν ήδη s δεδομένα, τότε επιλέγεται τυχαία ένα από αυτά και απορρίπτεται, ώστε το καινούριο δεδομένο να πάρει τη θέση του. (Leskovec, Rajaraman, Ullman, 2014)

2.2.2 Παράθυρο (Windowing)

Πρόκειται για μια διαδικασία τμηματοποίησης της ροής δεδομένων σε μικρά πακέτα ορισμένου μεγέθους σε συγκεκριμένο χρόνο. Όπως αναφέρθηκε παραπάνω τα δεδομένα συνεχούς ροής αλλάζουν στο χρόνο, οπότε είναι σημαντικό οι αλλαγές αυτές να αποτυπώνονται. Η προσέγγιση αυτή βοηθά στην αποτύπωση της μεταβολής της κατανομής των δεδομένων, καθώς το περιεχόμενο του παραθύρου αλλάζει δίνοντας τα ενημερωμένα δεδομένα. Υπάρχουν τέσσερις τύπους windowing: damped window (time fading), sliding window, landmark window, και tilted window (pyramidal) (Mansour, Abdullah, 2023).

Όλα τα Windows ενημερώνουν τα δεδομένα περιοδικά, ωστόσο διαφέρουν στη διαδικασία ενημέρωσης. Παρακάτω ακολουθούν οι τέσσερις μέθοδοι Windowing που αναφέρθηκαν.



Σχήμα 2-2: Τύποι Windowing (Mansour, Abdullah,2022)

- **Αποσβεσμένο Παράθυρο (Damped Window)**

Ονομάζεται και Time – Fading Window γιατί τα δεδομένα ξεθωριάζουν στο χρόνο. Αυτός ο τύπος παραθύρου δίνει συγκεκριμένη τιμή βάρους για κάθε δεδομένο που καταφθάνει. Τα καινούρια δεδομένα έχουν μεγαλύτερο βάρος, το οποίο στη συνέχεια μειώνεται αναλογικά με το χρόνο. Ουσιαστικά τα δεδομένα γερνάνε με την πάροδο του χρόνου και μειώνεται η σημαντικότητά τους, η οποία υπολογίζεται από την παρακάτω συνάρτηση:

$$F(x) = 2\lambda(t_c - t_0)$$

όπου λ ο παράγοντας ξεθωριάσματος t_0 η αρχική χρονική στιγμή και t_c η τρέχουσα χρονική στιγμή.

Όλα τα δεδομένα συμμετέχουν στη διαδικασία, ωστόσο έχουν διαφορετικό βάρος και άρα διαφορετικό επίπεδο σημαντικότητας. Μπορεί να θεωρηθεί μια ειδική περίπτωση Sliding Window ή Landmark Window.

- **Συρόμενο Παράθυρο (Sliding Window)**

Ο συγκεκριμένος τύπος παραθύρου είναι πιο κατάλληλος για εφαρμογές που αναφέρονται σε πιο πρόσφατες πληροφορίες, επειδή λαμβάνει υπόψιν μόνο τα πιο πρόσφατα δεδομένα. Έχει σταθερό μέγεθος, περιλαμβάνει δηλαδή συγκεκριμένο αριθμό δεδομένων και λειτουργεί με βάση την αρχή First-In-First-Out, δηλαδή τα παλιά δεδομένα αφαιρούνται και στη θέση τους προστίθενται τα νέα, ενώ το εύρος του παραθύρου παραμένει σταθερό. Όλα τα δεδομένα στο παράθυρο έχουν το ίδιο βάρος και άρα την ίδια σημαντικότητα. Τα παράθυρα μικρού μεγέθους δίνουν μεγαλύτερη ακρίβεια. Χρησιμοποιούνται επίσης όταν ο ακριβής αριθμός των σημείων δεδομένων είναι κρίσιμος για τη στατιστική ανάλυση, π.χ. παρακολούθηση της κυκλοφορίας.

- **Παράθυρο Ορόσημο (Landmark Window)**

Λειτουργεί τμηματοποιώντας τα δεδομένα ροής σε κομμάτια με βάση ορόσημα, τα οποία θα μπορούσαν να είναι συμβάντα δεδομένων ή ο χρόνος που έχει επέλθει. Το ορόσημο που ορίζεται ως σημείο εκκίνησης θα μπορούσε να είναι κάθε 100 σημεία, που σημαίνει ότι το παράθυρο ανανεώνεται λαμβάνοντας 100 νέα δεδομένα και απορρίπτοντας τα 100

προηγούμενα δεδομένα, τα οποία έχουν όλα το ίδιο βάρος. Ο συγκεκριμένος τύπος παραθύρου χρησιμοποιείται όταν απαιτούνται περιοδικά αποτελέσματα για τη στατιστική ανάλυση (ετήσια, μηνιαία, τριμηνιαία).

- **Παράθυρο Πυραμίδα (Pyramidal Window)**

Περιλαμβάνει δεδομένα από διαφορετικές χρονικές στιγμές, με διαφορετικά επίπεδα λεπτομερειών. Σε αντίθεση με το Sliding Window, το Pyramidal Window δεν απορρίπτει εντελώς τα παλιά δεδομένα, αλλά ενσωματώνει όλα τα δεδομένα, απλά δίνει μεγαλύτερο βάρος και άρα μεγαλύτερη σημαντικότητα στα πιο πρόσφατα.

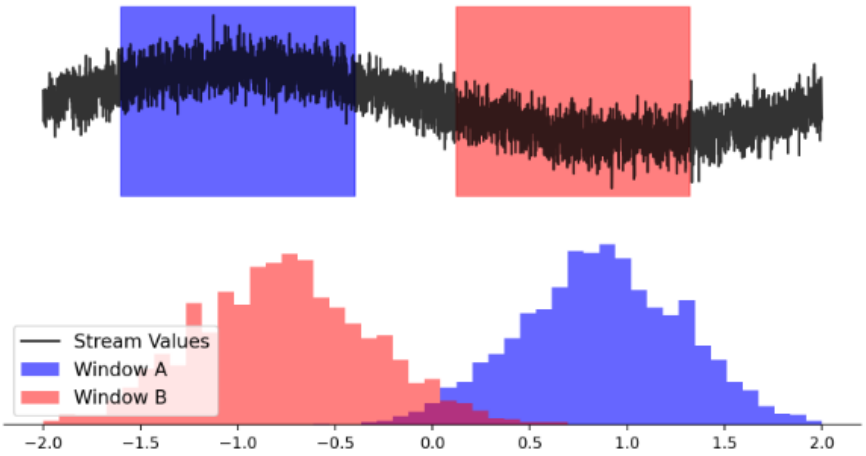
2.2.3 Ιστόγραμμα

Πρόκειται για ένα τύπο γραφήματος που έχει ευρείες εφαρμογές στη στατιστική. Παρέχει μια οπτική ερμηνεία των αριθμητικών δεδομένων υποδεικνύοντας το σύνολο των οποίων βρίσκεται μέσα σε ένα εύρος τιμών. Η συχνότητα των δεδομένων που εμπίπτουν σε κάθε κατηγορία, απεικονίζεται με τη χρήση ράβδου. Όσον αφορά στα δεδομένα συνεχούς ροής, αναπαριστά την σύνοψή τους, αποτυπώνοντας την κατανομή συχνοτήτων των τιμών που λαμβάνονται από τη ροή. Τα ιστογράμματα δεδομένων συνεχούς ροής χρειάζεται να ενημερώνονται συνεχώς, καθώς εισέρχονται νέα δεδομένα. Αυτό προκαλεί δυσκολίες, διότι ενδέχεται η μορφή τους να αλλάξει αρκετά μέσα σε ένα μικρό χρονικό διάστημα, αφού όπως έχει ήδη αναφερθεί, η κατανομή των δεδομένων αλλάζει και άρα και το εύρος τιμών τους. Για το λόγο αυτό κατά την κατασκευή των ιστογραμμάτων, θα πρέπει να δίνεται προτεραιότητα στα πιο πρόσφατα δεδομένα και να αγνοούνται τα παλαιότερα. Υπάρχουν δύο είδη ιστογραμμάτων συνεχούς ροής.

- **Ιστόγραμμα Παραθύρου (Windowed Histogram)**

Πρόκειται για ένα ιστόγραμμα που ακολουθεί την δομή του Sliding Window, περιλαμβάνει δηλαδή τα τελευταία n δεδομένα, όπου n ένας προκαθορισμένος αριθμός. Διαγράφοντας τα παλιά δεδομένα, το ιστόγραμμα προσαρμόζεται γρήγορα στις αλλαγές της κατανομής. Ωστόσο θα πρέπει τα n πρόσφατα δεδομένα να αποθηκεύονται πριν τη χρήση τους, οπότε δεσμεύεται

μνήμη. Για το λόγο αυτό είναι δύσκολο να δημιουργηθούν ιστογράμματα με μεγάλο αριθμό δεδομένων. Στο παρακάτω σχήμα φαίνονται τα ιστογράμματα που προκύπτουν για δύο διαφορετικά διαστήματα, με κόκκινο και μπλε, καθώς και οι τιμές που χρησιμοποιούνται για τη δημιουργία τους, με τα αντίστοιχα τετράγωνα.

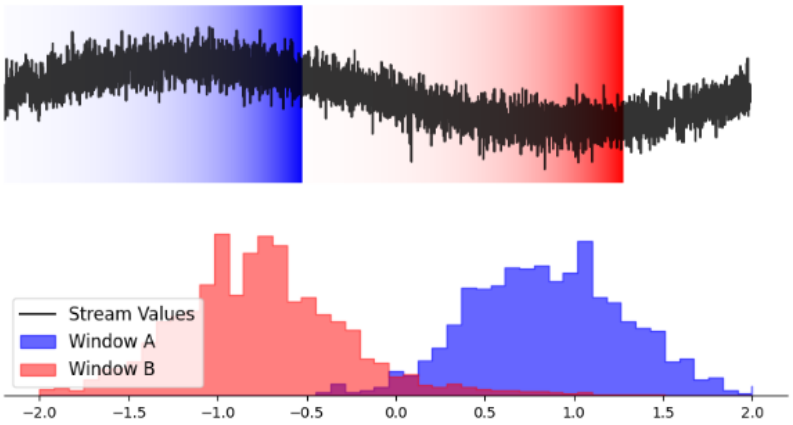


Σχήμα 2-3: Ιστόγραμμα Παραθύρου (Coleman,2022)

- **Ιστόγραμμα Εξασθένισης (Decayed Histogram)**

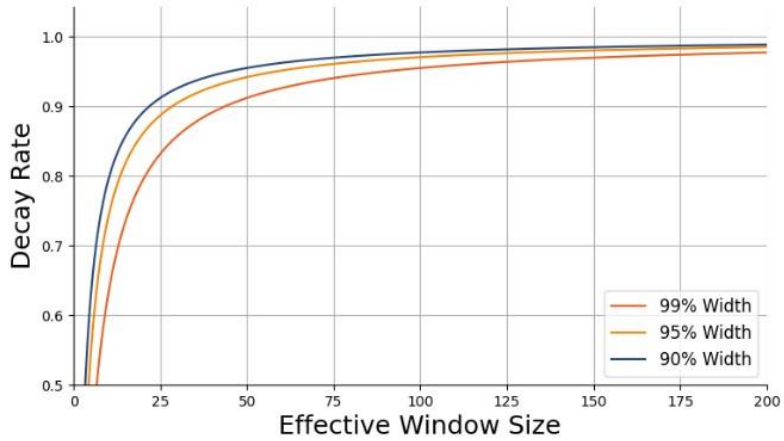
Η βασική ιδέα για την κατασκευή των ιστογραμμάτων αυτών είναι η σταδιακή εξασθένιση της συνεισφοράς των δεδομένων με την πάροδο του χρόνο, μέχρι να γίνει αμελητέα. Η εξασθένιση αυτή γίνεται εκθετικά. Πιο συγκεκριμένα κάθε φορά που ενημερώνεται το ιστόγραμμα, τα υπάρχοντα δεδομένα πολλαπλασιάζονται με έναν αριθμό μικρότερο του 1, τον πολλαπλασιαστή. Έτσι, κάθε φορά που προστίθεται μία νέα τιμή, μειώνεται πολλαπλασιαστικά η συνεισφορά των παλαιότερων δεδομένων. Στην περίπτωση αυτή δεν είναι τόσο ευκρινή τα όρια των δεδομένων που συμβάλουν στη δημιουργία του ιστογράμματος, καθώς υπάρχουν δεδομένα με σχεδόν μηδενική συνεισφορά, ωστόσο όχι αμελητέα.

Στο παρακάτω σχήμα φαίνονται τα ιστογράμματα που προκύπτουν σε δύο διαφορετικές χρονικές στιγμές. Όπως και πριν στα αντίστοιχα τετράγωνα αναπαρίστανται τα δεδομένα που συμβάλουν στην δημιουργία των δύο ιστογραμμάτων. Γίνεται εύκολα αντιληπτή η εξασθένιση της επιρροής των παλαιότερων δεδομένων, που απεικονίζεται με εξασθένιση των χρωμάτων.



Σχήμα 2-4: Ιστόγραμμα Εξασθένισης (Coleman,2022)

Όπως αναφέρθηκε, τα όρια των δεδομένων που συμπεριλαμβάνονται στο ιστόγραμμα δεν είναι ευκρινή. Θα πρέπει λοιπόν, να βρεθεί η τιμή της εξασθένισης, ώστε κάθε φορά να περιλαμβάνεται στο ιστόγραμμα συγκεκριμένος αριθμός δεδομένων (έστω W), δηλαδή το μεγαλύτερο ποσοστό του διαγράμματος να διαμορφώνεται από τα πιο πρόσφατα δεδομένα. Παρακάτω φαίνεται πώς επηρεάζεται η τιμή εξασθένισης από το πλήθος των δεδομένων. Πιο συγκεκριμένα, φαίνεται ο ρυθμός εξασθένισης που απαιτείται (άξονας x), για συγκεκριμένο μέγεθος παραθύρου (άξονας y), σε συγκεκριμένο επίπεδο στατιστικής σημαντικότητας. Για παράδειγμα το 95% των συνεισφορών προέρχεται από σημεία εντός του παραθύρου, και το 5% από εξωτερικά σημεία.



2-5: Σχέση τιμής εξασθένισης και πλήθους δεδομένων στο Ιστόγραμμα (Coleman, 2022)

Η συνολική συνεισφορά των δεδομένων που μπορούν να συμπεριληφθούν στο Ιστόγραμμα δίνεται από τον παρακάτω τύπο:

$$\sum_{t=0}^{T} \alpha^t$$

Καθώς το T μεγαλώνει και πηγαίνει στο άπειρο, η συνολική συνεισφορά δίνεται από την παρακάτω γεωμετρική σειρά:

$$\lim_{T\to\infty} \sum_{t=0}^{T} \alpha^t = \frac{1}{1-\alpha}$$

Συνεπώς σε μία ροή με άπειρα δεδομένα, η συνολική συνεισφορά των δεδομένων θα φτάσει στη σταθερή αυτή τιμή, λόγω σύγκλισης της ακολουθίας. Παρακάτω εξετάζεται η περίπτωση το άθροισμα να περιοριστεί μόνο στα πιο πρόσφατα σημεία W.

$$\sum_{n=0}^{W} \alpha^n = \frac{1-\alpha^W}{1-\alpha}$$

Τιμή ενδιαφέροντος αποτελεί το W, δηλαδή ο αριθμός των όρων που απαιτούνται ώστε το μερικό άθροισμα να αντιπροσωπεύει το μεγαλύτερο μέρος του πλήρους αθροίσματος - αυτά είναι τα σημεία που συμβάλλουν περισσότερο στην τιμή εξόδου και επομένως είναι τα σημεία που πρέπει να ληφθούν υπόψη "μέσα στο παράθυρο".

Για παράδειγμα, έστω ότι ο πολλαπλασιαστής παίρνει την τιμή 0.99 και το μερικό άθροισμα είναι 0.95. Παρακάτω υπολογίζεται ο αριθμός των όρων που θα συμμετέχουν στο ιστόγραμμα λύνοντας ως προς W.

$$\frac{1-0.99^W}{1-0.99} = \frac{0.95}{1-0.99} \Rightarrow W = \frac{log0.05}{log0.99} = 298$$

Άρα το 95% του μεγέθους του παραθύρου είναι 298 δεδομένα, με πολλαπλασιαστή α=0.99. Με άλλα λόγια, οι είσοδοι που είναι παλαιότερες από 298 βήματα αξίζουν συλλογικά μόνο το δ=5% της συνολικής μάζας στο ιστόγραμμα. Γενικότερα:

$$\frac{1-\alpha^W}{1-\alpha} = \frac{\delta}{1-\alpha} \quad \kappa\alpha\iota \quad \alpha = \delta^{\frac{1}{W}}$$

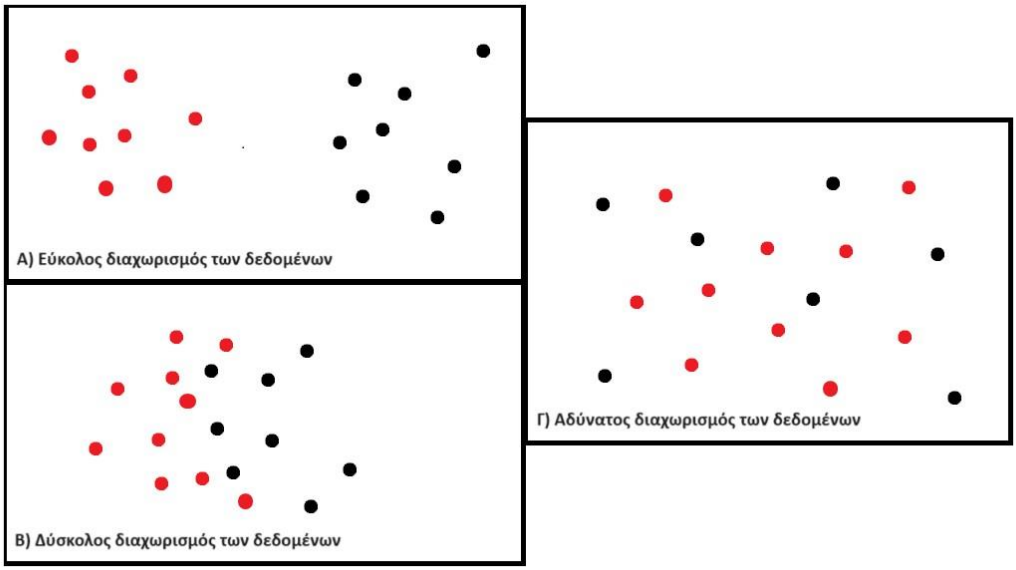
ΚΕΦΑΛΑΙΟ 3

Τελική Επεξεργασία

3.1 Συσταδοποίηση

Η συσταδοποίηση αποτελεί μια μη επιβλεπόμενη διαδικασία μηχανικής μάθησης που τμηματοποιεί το σύνολο των δεδομένων σε συστάδες, με βάση την ομοιότητά τους. Συνεπώς δεδομένα που μοιάζουν μεταξύ τους κατατάσσονται στην ίδια συστάδα, ενώ δεδομένα από άλλες ομάδες διαφέρουν σημαντικά μεταξύ τους.

Για αριθμητικά δεδομένα οι συστάδες μπορούν να αναπαρασταθούν ως νέφη σημείων στον *N*-διάστατο χώρο. Θα πρέπει να τονιστεί πως πολλές φορές τα δεδομένα δεν μπορούν εύκολα να διαχωριστούν καθώς δεν είναι ευκρινείς οι διαφορές τους.



Σχήμα 3-1: Νέφη Συστάδων

Το βασικό μέλημα της συσταδοποίησης είναι να αποκαλύψει την οργάνωση σε συστάδες, που επιτρέπουν την ανακάλυψη ομοιοτήτων αλλά και διαφορών, όπως επίσης και την εξαγωγή

χρήσιμων συμπερασμάτων για τα δεδομένα. Το κάθε κριτήριο που χρησιμοποιείται μπορεί να οδηγήσει σε διαφορετικές τμηματοποιήσεις, για το λόγο αυτό είναι σημαντικό να γίνει προεπεξεργασία των δεδομένων πριν την εφαρμογή της συσταδοποίησης. Επιπλέον θα πρέπει να βρεθεί ο τύπος των κλάσεων που αναζητούνται, καθώς ο κάθε αλγόριθμος είναι αποτελεσματικός για συγκεκριμένες διατάξεις των δεδομένων.

Η συσταδοποίηση αποτελεί ένα πολύ χρήσιμο εργαλείο για δεδομένα μεγάλου όγκου και βρίσκει εφαρμογή σε πολλά πεδία, τόσο στο χώρο των επιστημών όσο και στον χώρο των επιχειρήσεων. Αρχικά συμβάλει στη συμπίεση της πληροφορίας των δεδομένων, καθώς επιτρέπει να υιοθετηθούν ορισμένοι αντιπρόσωποι από κάθε συστάδα, αντί να διατηρείται το σύνολο των αρχικών δεδομένων. Χρησιμοποιείται επίσης για την παραγωγή υποθέσεων, μελετώντας τα όμοια στοιχεία που απαρτίζουν την κάθε συστάδα, τις οποίες και ελέγχει για την εγκυρότητά τους. Για παράδειγμα εξετάζοντας δεδομένα υγείας, σχηματίζεται η υπόθεση πώς οι ασθενείς με υψηλή πίεση εμφανίζουν καρδιακά προβλήματα. Αν πραγματοποιηθεί συσταδοποίηση σε δεδομένα που αφορούν πίεση και εμφάνιση καρδιακού νοσήματος και παρατηρηθεί ότι ασθενείς που πληρούν και τις δύο ιδιότητες ανήκουν στην ίδια ομάδα, η υπόθεση έχει επαληθευτεί.

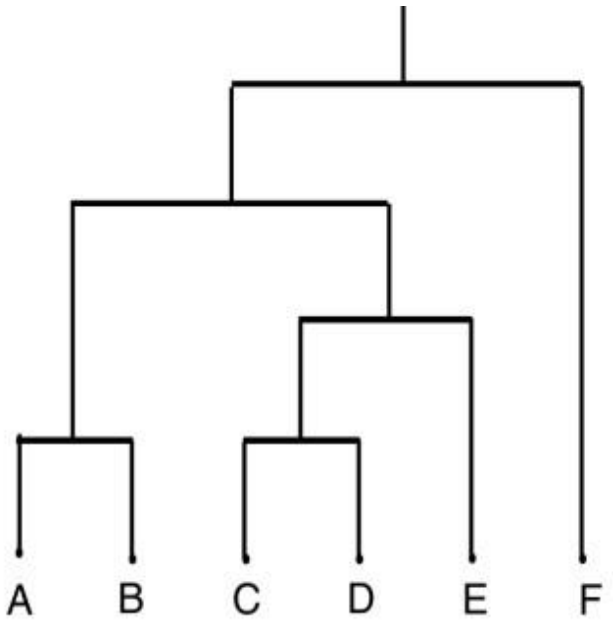
Τέλος μπορεί να πραγματοποιηθεί πρόβλεψη βασιζόμενη σε συστάδες. Πιο συγκεκριμένα να εφαρμοστεί συσταδοποίηση και να μελετηθούν τα μοτίβα και οι ομοιότητες των δεδομένων στην κάθε συστάδα. Στη συνέχεια, νέα άγνωστα δεδομένα να ταξινομηθούν στις συστάδες αυτές. Για παράδειγμα, έστω ότι η διαδικασία συσταδοποίησης εφαρμόζεται σε ένα σύνολο δεδομένων που αφορούν σε ασθενείς από την ίδια νόσο και το αποτέλεσμα είναι διάφορες ομάδες ασθενών, σύμφωνα με την αντίδρασή τους σε συγκεκριμένα φάρμακα. Για κάθε νέο ασθενή προσδιορίζεται η συστάδα στην οποία μπορεί να κατηγοριοποιηθεί και με τον τρόπο αυτό καθορίζεται η φαρμακευτική του αγωγή (Χακίδη, Βαζιργιάννης, 2005).

3.2 Αλγόριθμοι Συσταδοποίησης

Η αλγόριθμοι συσταδοποίησης χωρίζονται σε πέντε κατηγορίες ανάλογα με τον τρόπο που λειτουργούν. Στους Ιεραρχικούς, στους Διαμεριστικούς, σε αυτούς που ομαδοποιούν με βάση την πυκνότητα, με βάση το πλέγμα και με βάση το μοντέλο. (Muhammad, 2024)

1. Ιεραρχικοί Αλγόριθμοι

Στους Ιεραρχικούς αλγόριθμους συσταδοποίησης, κάθε επίπεδο αντιπροσωπεύει ένα σύνολο από συστάδες και το αποτέλεσμα απεικονίζεται σε ένα δενδρόγραμμα. Πρόκειται για μία δεντρική δομή όπου κάθε στοιχείο απεικονίζεται ως φύλλο στο τελευταίο επίπεδο και αποτελεί μια ξεχωριστή συστάδα. Στα προηγούμενα επίπεδα τα φύλλα ενώνονται και στο πρώτο επίπεδο, στη ρίζα, όλα τα στοιχεία αποτελούν μία συστάδα, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 6: Δενδρόγραμμα

Οι ιεραρχικοί αλγόριθμοι χωρίζονται σε δύο επιπλέον κατηγορίες, τους συσσωρευτικούς (agglomerative) και τους διαιρετικούς (divisive). Στους συσσωρευτικούς αρχικά κάθε στοιχείο είναι μια συστάδα και επαναληπτικά οι συστάδες συγχωνεύονται. Στους διαιρετικούς αλγορίθμους στην αρχή όλα τα στοιχεία αποτελούν μία συστάδα και προοδευτικά διαιρούνται. Οι ιεραρχικοί αλγόριθμοι έχουν υψηλή πολυπλοκότητα και είναι ευαίσθητοι στις ακραίες τιμές.

2. Διαμεριστικοί Αλγόριθμοι

Οι διαμεριστικοί αλγόριθμοι χωρίζουν τα δεδομένα σε ένα προκαθορισμένο σύνολο συστάδων, με βάση την ομοιότητα των δεδομένων μεταξύ τους, ή την απόσταση τους από τα κέντρα των συστάδων. Οι συστάδες που δημιουργούνται έχουν αυστηρά σφαιρικό σχήμα, ενώ το κέντρο τους βελτιώνεται επαναληπτικά. Πιο συγκεκριμένα το αρχικό σύνολο των κέντρων των συστάδων επιλέγεται τυχαία και στη συνέχεια τα στοιχεία μετακινούνται μεταξύ των συστάδων μέχρι να επιτευχθεί ισορροπία. Οι διαμεριστικοί αλγόριθμοι χρησιμοποιούνται ευρέως λόγω της ταχύτητάς τους, αλλά και της δυνατότητας να εφαρμοστούν σε μεγάλα σύνολα δεδομένων.

3. Αλγόριθμοι συσταδοποίησης με βάση την πυκνότητα

Οι αλγόριθμοι συσταδοποίησης με βάση την πυκνότητα εντοπίζουν συστάδες αυθαίρετου σχήματος και ανιχνεύουν τον αριθμό των συστάδων χωρίς να χρειάζεται να προκαθοριστεί από τον χρήστη. Οι συστάδες αποτελούνται από πυκνές περιοχές και χωρίζονται από περιοχές χαμηλής πυκνότητας, οι οποίες περιέχουν ακραίες τιμές και θορύβους. Πρόκειται για αλγόριθμους ανθεκτικούς στο θόρυβο, ωστόσο πρέπει να επιλεχθούν πολλοί παράμετροι.

4. Αλγόριθμοι συσταδοποίησης με βάση το πλέγμα

Οι αλγόριθμοι συσταδοποίησης με βάση το πλέγμα είναι αποτελεσματικοί στην εξόρυξη μεγάλων πολυδιάστατων συνόλων δεδομένων. Χωρίζουν το χώρο δεδομένων σε έναν πεπερασμένο αριθμό κελιών για να σχηματίσουν μια δομή πλέγματος και στη συνέχεια

δημιουργούν κλάσεις από τα κελιά αυτά. Οι κλάσεις αντιστοιχούν σε περιοχές που είναι πιο πυκνές σε σημεία δεδομένων από το περιβάλλον τους.

Το μεγάλο πλεονέκτημα της συσταδοποίησης βάσει πλέγματος είναι η σημαντική μείωση της χρονικής πολυπλοκότητας, ειδικά για πολύ μεγάλα σύνολα δεδομένων. Αντί να ομαδοποιούν απευθείας τα σημεία δεδομένων, οι αλγόριθμοι που βασίζονται σε πλέγμα συγκεντρώνουν τα σημεία δεδομένων που αντιπροσωπεύονται από κελιά. Στις περισσότερες εφαρμογές, καθώς ο αριθμός των κελιών είναι σημαντικά μικρότερος από τον αριθμό των σημείων δεδομένων, η απόδοση βελτιώνεται σημαντικά. Οι αλγόριθμοι ομαδοποίησης που βασίζονται σε πλέγμα συνήθως περιλαμβάνουν τα ακόλουθα πέντε βήματα:

Βήμα 1 Δημιουργία της δομής πλέγματος, δηλ. διαμερισμός του χώρου δεδομένων σε έναν πεπερασμένο αριθμό κελιών.

Βήμα 2 Υπολογισμός της πυκνότητας κελιών για κάθε κελί.

Βήμα 3 Ταξινόμηση των κελιών ανάλογα με την πυκνότητά τους.

Βήμα 4 Προσδιορισμός κέντρων κλάσης.

Βήμα 5 Διέλευση γειτονικών κελιών.

Δεδομένου ότι η πυκνότητα κελιών χρειάζεται συχνά να υπολογιστεί για την ταξινόμηση κελιών και την επιλογή κέντρων, οι περισσότεροι αλγόριθμοι συσταδοποίησης που βασίζονται σε πλέγμα μπορούν επίσης να θεωρηθούν βασισμένοι στην πυκνότητα. Ορισμένοι αλγόριθμοι συσταδοποίησης που βασίζονται σε πλέγμα συνδυάζουν επίσης ιεραρχική συσταδοποίηση προκειμένου να οργανώσουν κελιά με βάση την πυκνότητά τους. (Mingjing, Fuyu, 2022).

5. Αλγόριθμοι συσταδοποίησης με βάση το μοντέλο

Οι αλγόριθμοι αυτοί, εντοπίζουν την κατανομή στην οποία προσαρμόζονται καλύτερα τα δεδομένα. Όπως είναι λογικό η απόδοση τους εξαρτάται άμεσα από το μοντέλο. Ένα από τα σημαντικότερα πλεονεκτήματα των αλγορίθμων αυτών είναι η ανθεκτικότητα στο θόρυβο.

Στον παρακάτω πίνακα συνοψίζονται τα πλεονεκτήματα και τα μειονεκτήματα της κάθε κατηγορίας.

ΑΛΓΟΡΙΘΜΟΣ	ΠΛΕΟΝΕΚΤΗΜΑΤΑ	ΜΕΙΟΝΕΚΤΗΜΑΤΑ
ΙΕΡΑΡΧΙΚΟΣ	• Οργανωμένη Δομή εξόδου (Δενδρόγραμμα)	• Υψηλή πολυπλοκότητα • Ευαισθησία στα outliers
ΔΙΑΜΕΡΙΣΤΙΚΟΣ	• Εύκολη εκτέλεση	• Συστάδες σφαιρικού σχήματος • Προκαθορισμένο πλήθος συστάδων
ΒΑΣΕΙ ΠΥΚΝΟΤΗΤΑΣ	• Αυθαίρετο σχήμα • Ανθεκτικότητα στο θόρυβο	• Προκαθορισμένες παράμετροι • Κλάσεις διαφορετικής πυκνότητας
ΒΑΣΕΙ ΠΛΕΓΜΑΤΟΣ	• Αυθαίρετο σχήμα • Γρήγορη εκτέλεση	• Μόνο για δεδομένα μικρής διάστασης • Προκαθορισμένο μέγεθος πλέγματος
ΒΑΣΕΙ ΜΟΝΤΕΛΟΥ	• Ανθεκτικότητα στο θόρυβο	• Μεγάλη εξάρτηση από το μοντέλο

Πίνακας 3-1: Πλεονεκτήματα και Μειονεκτήματα αλγορίθμων συσταδοποίησης

3.3 Αλγόριθμος DBIECM

Ως μέθοδος συσταδοποίησης θα χρησιμοποιηθεί αλγόριθμος DBIECM. Πρόκειται για έναν Διαμεριστικό αλγόριθμο, ο οποίος επιλέγεται για τις ανάγκες της παρούσας Διπλωματικής λόγω της απλότητας, της εύκολης εκτέλεσής και της ταχύτητας του. Επιπλέον όπως αναφέρθηκε και παραπάνω οι διαμεριστικοί αλγόριθμοι ενδείκνυνται για μεγάλα σύνολα δεδομένων. Προτού όμως αναλυθεί θα πρέπει να οριστούν ο αλγόριθμος ECM, καθώς επίσης και το μέτρο αξιολόγησης DBI.

Η μέθοδος εξελιγμένης συσταδοποίησης (Evolving Clustering Method – ECM) είναι μία γρήγορη μέθοδος διαχωριστικής συσταδοποίησης που προσαρμόζεται εύκολα στα δεδομένα συνεχούς ροής και πραγματοποιεί μόνο ένα πέρασμα σε καθένα από αυτά. Πιο συγκεκριμένα, καθώς ο αλγόριθμος εκτελείται, ασχολείται μόνο με τα νέα δεδομένα, εξοικονομώντας έτσι

χρόνο επεξεργασίας. Βασίζεται στην απόσταση και ζητάει από το χρήστη να ορίσει ένα όριο ($Dthr$) για την μέγιστη ακτίνα των συστάδων που θα δημιουργήσει. Ως εκ τούτου, οι συστάδες θα έχουν σφαιρικό σχήμα με ακτίνα μικρότερη ή ίση με το όριο $Dthr$. Χρησιμοποιώντας το $Dthr$ αποφασίζει αν τα δεδομένα ανήκουν σε κάποια από τις υπάρχουσες συστάδες, διαφορετικά δημιουργεί μία καινούργια. Στην αρχή τα κέντρα των συστάδων ορίζονται τυχαία. Με τη διέλευση νέων δεδομένων, τα κέντρα των συστάδων αλλάζουν, όπως επίσης αλλάζει και η ακτίνα της συστάδας που απαρτίζουν, μέχρι να φτάσει το όριο $Dthr$.

Ο δείκτης DBI (Davies-Bouldin Index – DBI) αποτελεί ένα κριτήριο αξιολόγησης που βασίζεται στην ομοιότητα του εσωτερικού της συστάδες, αλλά και στη διαφορά μεταξύ των συστάδων. Κατά τη διαδικασία της συσταδοποίησης, η αποτελεσματικότητα της υπολογίζεται από το βαθμό ομοιότητας των δεδομένων μέσα στη συστάδα και το βαθμό διαφοράς των δεδομένων στις διαφορετικές συστάδες. Ο τύπος υπολογισμού του DBI είναι ο ακόλουθος.

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{\substack{j\neq i \\ j=1,2,\dots k}} \left\{\frac{S_i + S_j}{d(Cc_i, Cc_j)}\right\}$$

Με $d(Cc_i, Cc_j)$ συμβολίζεται η απόσταση μεταξύ των συστάδων Cc_i και Cc_j. Με S_i συμβολίζεται το τυπικό σφάλμα της απόστασης του i-οστού δεδομένου της συστάδας από το κέντρο της συστάδας Cc_i, το οποίο εκφράζει την ομοιότητα εντός της i-οστής συστάδας. Ο τύπος του S_i είναι ο ακόλουθος.

$$S_i = \sqrt{\frac{\sum_{j=1}^{m_j} |x_j - Cc_i|^2}{m_i}}$$

με m_i συμβολίζεται ο αριθμός των δεδομένων στην i-οστή συστάδα.

Καθώς η ομοιότητα των δεδομένων εντός της συστάδας και οι διαφορές τους με εκείνα που έχουν ταξινομηθεί σε άλλες συστάδες αυξάνονται, το μέτρο DBI μειώνεται και η ποιότητα της συσταδοποίησης βελτιώνεται.

Προχωρώντας, ο Αλγόριθμος DBIECM είναι μια βελτιωμένη εκδοχή του ECM που βασίζεται επίσης στην απόσταση και απαιτεί από το χρήστη να ορίσει την μέγιστη ακτίνα συστάδας. Με τη διέλευση ενός νέου στοιχείου $\boldsymbol{x}$, γίνεται προσπάθεια από το αλγόριθμο να

ενταχθεί σε μία υπάρχουσα συστάδα. Για το σκοπό αυτό, υπολογίζονται οι αποστάσεις μεταξύ του $\boldsymbol{x}$ και όλων των κέντρων των συστάδων. Εφόσον η ακτίνα κάποιας συστάδας είναι μεγαλύτερη ή ίση από την απόστασή της από το $\boldsymbol{x}$, τότε το νέο στοιχείο προστίθεται σε αυτήν. Αξίζει να σημειωθεί πως το $\boldsymbol{x}$ μπορεί να προστεθεί σε παραπάνω από μία συστάδες, αρκεί να πληροί την παραπάνω προϋπόθεση. Σε περίπτωση που η απόσταση κάθε συστάδας από το $\boldsymbol{x}$ είναι μεγαλύτερη από τις ακτίνες τους και οι ακτίνες αυτές είναι ίσες με τη μέγιστη τιμή που έχει ορίσει ο χρήστης, τότε δημιουργείται μία νέα συστάδα που περιέχει το $\boldsymbol{x}$.

Θα πρέπει να τονιστεί πως η παράμετρος της μέγιστης ακτίνας που ορίζεται από το χρήστη επηρεάζει σε μεγάλο βαθμό το αποτέλεσμα της συσταδοποίησης και μία διαφορετική τιμή ενδέχεται να οδηγήσει σε τελείως διαφορετικές συστάδες. Ο ορισμός της ακτίνας εξαρτάται από τα δεδομένα και για το λόγο αυτό απαιτείται προσεκτική μελέτη και εξοικείωση με αυτά.

Όσον αφορά στην ανάλυση πολυπλοκότητας, με την έλευση ενός στοιχείου x εκτελείται γραμμική ανζήτηση στις συστάδες για να διαπιστωθεί αν η απόσταση από το κέντρο τους είναι μικρότερη από την ακτίνα τους. Με τον όρο γραμμική αναζήτηση, εννοείται ότι ο αλγόριθμος πραγματοποιέι ένα μόνο πέρασμα στα k στοιχεία. Για το λόγο αυτό η πολυπλοκότητα της γραμμικής αναζήτησης είναι $O(k)$. Επιπλέον για τον υπολογισμό του μέτρου αξιολόγησης DBI απαιτείται ο υπολογισμός των αποστάσεων μεταξύ των συστάδων. Συνεπώς έχουμε δύο περάσματα για κάθε συστάδα και άρα η πολυπλοκότητα είναι $O(k^2)$. Ωστόσο όταν το νέο δεδομένο μπορεί να ταξινομηθεί σε περισσότερες από μία κλάσεις, τοποθετείται σε όλες και υπολογίζεται το DBI. Οι υπολογισμοί που απαιτούνται έχουν πολυπλοκότητα $O(k^3)$. Άρα στην χειρότερη περίπτωση η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(k^3)$ (Zhang, Zhong, Tian, Zhang,Li,2017).

Στη συνέχεια δίνεται η δομή του αλγορίθμου.

Η είσοδος του αλγορίθμου είναι τα δεδομένα $\boldsymbol{x}_i$ και το όριο $Dthr$, ενώ η έξοδος είναι τα αποτελέσματα της συσταδοποίησης. Ο αλγόριθμος μπορεί να υλοποιηθεί με βάση τα ακόλουθα βήματα.

Βήμα 1: Δημιουργία της πρώτης συστάδας C_1 χρησιμοποιώντας το x_1 ως κέντρο της. Η ακτίνα της πρώτης συστάδας είναι 0

Βήμα 2: Υπολογισμός της απόστασης του νέου δεδομένου x_i από τις ακτίνες όλων των συστάδων Cc_j

$D_{ij} = \|x_i - Cc_j\|, j = 1,2, \dots k$, όπου k είναι ο αριθμός των συστάδων που υπάρχουν.

Σε περίπτωση που έχουν ταξινομηθεί όλα τα δεδομένα ο αλγόριθμος σταματάει.

Βήμα 3: Τα υπάρχοντα δεδομένα έχουν ήδη ταξινομηθεί σε κάποια κλάση. Συνεπώς κάθε x_i ανήκει σε μία συστάδα με κέντρο Cc_m, όπου

$$D_{im} = \|x_i - Cc_m\| = \min_{j=1,2,\dots,k}(\|x_i - Cc_j\|)$$

Με την προσθήκη ενός νέου δεδομένου σε μία συστάδα, ανανεώνεται το κέντρο της. Το νέο κέντρο υπολογίζεται ως εξής

$$Cc_j = \frac{1}{k}\sum_{i=1}^{m_j} x_i$$

Ανανεώνεται επίσης η ακτίνα της συστάδας, η οποία υπολογίζεται από τον παρακάτω τύπο

$$R_{uj} = \max_{i=1,2,\dots,k}\{d(x_i, Cc_j)\}$$

Βήμα 4: Αν $D_{ij} > Dthr$, τότε το x_i δεν ανήκει σε κάποια από τις υπάρχουσες συστάδες, συνεπώς δημιουργείται μία νέα συστάδα με κέντρο το x_i. Στη συνέχεια εφαρμόζονται τα Βήματα 1 και 2.

Βήμα 5: Εφόσον υπάρχει κάποια συστάδα για την οποία ισχύει

$R_{uj} < D_{ij} < Dthr,\ j = 1,2,..,k$, όπου $Dthr$ το μέγιστο όριο της ακτίνας των συστάδων,

Τότε το x_i προστίθεται στην συστάδα αυτή και ανανεώνεται το κέντρο και η ακτίνα της.

Προκειμένου να επιλεχθεί το κατάλληλο όριο *Dthr*, δηλαδή η κατάλληλη μέγιστη ακτίνα, θα πρέπει να οριστεί ένα μέτρο που θα υπολογίζει το βέλτιστο δυνατό αποτέλεσμα συσταδοποίησης. Το Silhouette score είναι μία μετρική που χρησιμοποιείται για να αξιολογήσει την ποιότητα των συστάδων που δημιουργούνται από έναν αλγόριθμο συσταδοποίησης. Μετράει πόσο παρόμοιο είναι ένα στοιχείο με τα υπόλοιπα στοιχεία της συστάδας στην οποία ανήκει σε σύγκριση με άλλες συστάδες. Το Silhouette Score χρησιμοποιείται γενικότερα για την αξιολόγηση της απόδοσης αλγορίθμων ομαδοποίησης, όπως το k-means, η ιεραρχική ομαδοποίηση και το DBSCAN. Βοηθά στον προσδιορισμό του βέλτιστου αριθμού συστάδων συγκρίνοντας τις βαθμολογίες Silhouette για διαφορετικούς αριθμούς συστάδων. Η βαθμολογία κυμαίνεται από -1 έως 1, με τις υψηλότερες τιμές να υποδηλώνουν καλύτερα καθορισμένες και πιο κατάλληλες συστάδες.

Ο συντελεστής Silhouette $s(i)$ για ένα μεμονωμένο δείγμα i ορίζεται ως εξής

$$s(i) = \frac{b - a}{\max(a, b)}$$

όπου:

a: Η μέση απόσταση μεταξύ ενός δείγματος και όλων των άλλων σημείων στην ίδια συστάδα (ενδοσυσταδική απόσταση).

b: Η μέση απόσταση μεταξύ ενός δείγματος και όλων των σημείων στην πλησιέστερη συστάδα (την συστάδα στην οποία το δείγμα δεν ανήκει, αλλά είναι πιο κοντά).

Αν το $s(i)$ είναι κοντά στο 1, το δείγμα είναι καλά ομαδοποιημένο. Αν το $s(i)$ είναι κοντά στο 0, το δείγμα βρίσκεται πολύ κοντά στο όριο απόφασης μεταξύ δύο γειτονικών συστάδων και συνεπώς γίνεται δύσκολη η ομαδοποίηση και ο διαχωρισμός. Αν το $s(i)$ είναι κοντά στο -1, το δείγμα μπορεί να έχει ανατεθεί στη λάθος συστάδα.

Η βαθμολογία Silhouette για ολόκληρο το σύνολο δεδομένων είναι ο μέσος συντελεστής Silhouette για όλα τα δείγματα, δηλαδή

$$Silhouette\ score = \frac{1}{N}\sum_{i=1}^{N} s\,(i)$$

όπου N είναι ο συνολικός αριθμός δειγμάτων.

Η ερμηνεία της βαθμολογίας Silhouette δίνεται παρακάτω:

- **Υψηλή Βαθμολογία Silhouette (κοντά στο 1)**: Υποδηλώνει ότι οι συστάδες είναι καλά διαχωρισμένες και κάθε δείγμα ταιριάζει καλά στη δική του συστάδα.
- **Χαμηλή Βαθμολογία Silhouette (γύρω στο 0)**: Υποδηλώνει ότι οι συστάδες δεν είναι καλά διαχωρισμένες και τα δείγματα είναι κοντά στο όριο απόφασης μεταξύ συστάδων.
- **Αρνητική Βαθμολογία Silhouette**: Υποδηλώνει ότι τα δείγματα μπορεί να βρίσκονται στη λάθος συστάδα (Ketan, Charles, 2020).

3.4 Κανόνες Συσχέτισης

Οι κανόνες συσχέτισης αποτελούν μία σύγχρονη μέθοδο για την εξαγωγή γνώσης δεδομένων από μεγάλες βάσεις δεδομένων, που συγκεντρώνουν και περιγράφουν σημαντικές πληροφορίες σε διάφορους τομείς. Αρχικά εμφανίστηκαν για την ανάλυση του «καλαθιού της αγοράς», καθώς στις υπεραγορές, συγκεντρωνόταν τεράστιος όγκος δεδομένων σχετικά με τις αγορές των πελατών. Προέκυψε έτσι η ανάγκη για αξιοποίηση και μελέτη της πληροφορίας αυτής και δημιουργήθηκαν οι κανόνες συσχέτισης.

Χαρακτηριστικό παράδειγμα αποτελεί και ο ακόλουθος κανόνας: «Οι πελάτες που αγοράζουν γάλα, αγοράζουν συγχρόνως και ψωμί σε ποσοστό 60%». Η πρόταση αυτή, συνδέει το αίτιο (αγορά γάλατος), με το αποτέλεσμα (αγορά ψωμιού), παρέχοντας μία ένδειξη για το πόσο πιθανό είναι να υπάρχει μία τέτοια σχέση αιτίας-αιτιατού.

Οι κανόνες συσχέτισης βρήκαν πεδίο εφαρμογής σε διάφορες πτυχές της αγοράς. Αρχικά η σχέση αιτίας αποτελέσματος επηρέασε άμεσα την προώθηση των προϊόντων. Για παράδειγμα, έστω ο κανόνας μακαρόνια−>τυρί, που σημαίνει πώς οι πελάτες που αγοράζουν μακαρόνια, αγοράζουν συγχρόνως και τυρί σε ένα ποσοστό. Το γεγονός ότι τα μακαρόνια βρίσκονται στο πρώτο μέλος, μπορεί να χρησιμοποιηθεί ώστε να αυξηθούν οι πωλήσεις στο δεύτερο μέλος, δηλαδή στο τυρί. Παράλληλα καθώς οι κανόνες συσχέτισης υποδηλώνουν τις τάσεις της αγοράς, επηρεάζουν και τον τρόπο που τοποθετούνται τα προϊόντα στα ράφια. Έτσι, ο πελάτης

διευκολύνεται και παρακινείται να αγοράσει τα συσχετιζόμενα προϊόντα. Τέλος, η μελέτη της τάσης με την οποία διακινούνται τα προϊόντα βοηθά στην οργάνωση και διαχείριση των αποθεμάτων.

Για να οριστούν οι κανόνες συσχέτισης, θα πρέπει αρχικά να οριστούν δύο σύνολα. Έστω $I = \{i_1, i_2, \ldots, i_m\}$ ένα σύνολο από διακριτά αντικείμενα (items) και D ένα σύνολο από δοσοληψίες (transactions). Κάθε δοσοληψία T είναι ένα σύνολο από αντικείμενα για τα οποία ισχύει $T \subseteq I$. Κανόνας συσχετισης είναι μία συσχέτιση της μορφής $X \rightarrow Y$, όπου $X \subseteq I, \quad Y \subseteq I$ και $X \cap Y = \emptyset$. Το πρώτο μέλος του κανόνα καλείται υπόθεση και το δεύτερο συμπέρασμα. Ο κανόνας $X \rightarrow Y$ ισχύει στις δοσοληψίες D με εμπιστοσύνη (confidence) c, αν το $c\%$ των δοσοληψιών που περιέχουν το X περιέχουν επίσης και το Y ή αλλιώς η δεσμευμένη πιθανότητα $P(Y|X) = \frac{P(X \cup Y)}{P(X)} = c$. Ο κανόνας $X \rightarrow Y$ έχει υποστήριξη (support) s, αν το $s\%$ των δοσοληψιών στο D περιέχουν το $X \cup Y$, ή αλλιώς $P(X \cup Y) = s$. Συνεπώς, η υποστήριξη δείχνει πόσο συχνά εμφανίζονται τα αντικείμενα του κανόνα και η εμπιστοσύνη δίνει την ισχύ συνεπαγωγής του κανόνα. Μια υψηλή υποστήριξη του κανόνα, είναι ένδειξη πώς ένας μεγάλος αριθμός εγγραφών περιέχει τόσο το αριστερό όσο και το δεξί μέλος του κανόνα αυτού και έτσι μπορεί να θεωρηθεί ως αντιπροσωπευτικός κανόνας του συνόλου δεδομένων.

Είναι προφανές ότι οι κανόνες $X \rightarrow Y$ και $Y \rightarrow X$ έχουν την ίδια υποστήριξη, αλλά χρησιμότερος είναι ο κανόνας με τη μεγαλύτερη εμπιστοσύνη, καθώς είναι ισχυρότερη η σχέση αιτίας και αποτελέσματος (Χακίδη, Βαζιργιάννης, 2005)

3.5 Κανόνες Συσχέτισης σε Δεδομένα Συνεχούς Ροής

Όσον αφορά στα δεδομένα συνεχούς ροής, τα πράγματα είναι λίγο πιο σύνθετα. Προκειμένου να χρησιμοποιηθούν τεχνικές για κανόνες συσχέτισης, θα πρέπει να ληφθούν υπόψιν ορισμένα ζητήματα. Τα ζητήματα αυτά αναπτύσσονται παρακάτω.

Αρχικά, θα πρέπει να επιλεγεί το τμήμα των δεδομένων στο οποίο θα εφαρμοστούν οι κανόνες συσχέτισης. Αυτό μπορεί εύκολα να διευθετηθεί χρησιμοποιώντας Windowing. Πιο συγκεκριμένα, με τη χρήση του Landmark Window επιλέγονται τα δεδομένα, από μία

συγκεκριμένη στιγμή (ορόσημο). Με το Damped Window επιλέγονται τα δεδομένα, αρκεί να είναι πρόσφατα, καθώς η σημαντικότητά τους ξεθωριάζει με το χρόνο και με το Sliding Window επιλέγονται τα δεδομένα σε παράθυρα.

Το επόμενο κρίσιμο ζήτημα που προκύπτει, είναι η σωστή διαχείριση της περιορισμένης διαθέσιμης μνήμης κατά την εκτέλεση του αλγορίθμου. Αυτό περιλαμβάνει την διαδικασία επιλογής της πληροφορίας που θα συλλεχθεί από τη ροή δεδομένων, αλλά και την αποθήκευση ανανέωση και ανάκλησή της. Στα στατικά δεδομένα συλλέγεται το σύνολο της πληροφορίας και στη συνέχεια απορρίπτονται εκείνα που δεν είναι συχνά. Το παραπάνω δεν είναι εφικτό στα δεδομένα συνεχούς ροής, καθώς δεν υπάρχει αρκετή μνήμη ώστε να αποθηκευτεί η πληροφορία στο σύνολό της και το πλήθος των δεδομένων αλλάζει με το χρόνο, καθώς έρχονται συνεχώς νέα δεδομένα. Για το λόγο αυτό θα πρέπει να συλλεχθεί και να αποθηκευτεί σε σύντομο χρονικό διάστημα, όσον το δυνατόν λιγότερη πληροφορία, αλλά αρκετή ώστε να παραχθούν κανόνες συσχέτισης.

Το πρόβλημα απλοποιείται και ανάγεται στην εύρεση των πιο συχνών δεδομένων που εμφανίζονται στη ροή, με το βέλτιστο και αποδοτικότερο τρόπο. Επιθυμητό είναι για εξοικονόμηση χρόνου και μείωση της πολυπλοκότητας, η ανάλυση να γίνει με ένα πέρασμα στα δεδομένα (Jiang, Gruenwald, 2006)

3.5.1 Αλγόριθμοι για εύρεση συχνών στοιχείων

Οι αλγόριθμοι για εύρεση συχνών στοιχείων είναι τεχνικές που χρησιμοποιούνται στην εξόρυξη δεδομένων για τον εντοπισμό στοιχείων που εμφανίζονται μαζί σε ένα σύνολο δεδομένων, με μεγάλη συχνότητα. Αυτοί οι αλγόριθμοι είναι ιδιαίτερα χρήσιμοι για την εξαγωγή κανόνων συσχέτισης και για την ανάλυση των σχέσεων μεταξύ αντικειμένων.

Οι Richard M. Karp, Scott Shenker και Christos H. Papadimitriou δημιούργησαν έναν απλό αλγόριθμο για της εύρεση συχνών στοιχείων σε ροές δεδομένων. Ο αλγόριθμος ζητά από το χρήστη να ορίσει ένα ποσοστό $\theta \in (0,1)$ και στη συνέχεια υπολογίζει το πλήθος εμφάνισης όλων των στοιχείων. Αν το ποσοστό εμφάνισης κάποιου δεδομένου ξεπερνά το θ, τότε το

δεδομένο αυτό θεωρείται συχνό και αποθηκεύεται τόσο το ίδιο, όσο και το πλήθος εμφάνισής του. Το πρόβλημα με τον αλγόριθμο αυτό είναι ότι απορρίπτει δεδομένα τα οποία μπορεί να μην είναι συχνά εκείνη την στιγμή, αλλά να εμφανιστούν μαζικά σε επόμενη στιγμή. Υπάρχει δηλαδή πιθανότητα να απορρίψει δεδομένα που θα γίνουν συχνά στο μέλλον

Οι Baiyin ,Yang, Karen Watkins και Marsick δημιούργησαν έναν αλγόριθμο, που αποθηκεύει στην διαθέσιμη μνήμη το πλήθος εμφάνισης των δεδομένων εκείνων που δεν ξεπερνούν τις τρεις εμφανίσεις ($k \leq 3$, όπου k είναι το μέγιστο όριο των συχνών δεδομένων). Ωστόσο όπως είναι λογικό η εφαρμογή του αλγορίθμου αυτού μπορεί να γίνει σε πολύ περιορισμένες περιπτώσεις. Πιο συγκεκριμένα θα πρέπει $k \leq 3$ και $n \leq 1800$ (όπου n το σύνολο των διακεκριμένων δεδομένων). Είναι προφανές πώς όσο περισσότερα δεδομένα συλλέγονται, τόσο περισσότερη μνήμη δαπανάται, προκειμένου να πάρουμε έγκυρα αποτελέσματα. Απαιτείται επίσης περισσότερος χρόνος.

Λόγω της περιορισμένης μνήμης, για την αποθήκευση, την ενημέρωση και την ανάκτηση του τεράστιου όγκου δεδομένων, απαιτείται μία συμπαγής και αποτελεσματική δομή. Αποτυχία στην ανάπτυξη μίας τέτοιας δομής, οδηγεί στη μείωση της αποτελεσματικότητας του αλγορίθμου εξόρυξης, καθώς ακόμα και αν η πληροφορία αποθηκευτεί στο σύνολό της, θα αυξηθεί ο χρόνος επεξεργασίας της. Επιπλέον, η δομή δεδομένων θα πρέπει να διατηρείται, διότι δεν είναι δυνατή η επαναλαμβανόμενη σάρωση των δεδομένων από την αρχή. Ο αλγόριθμος δηλαδή θα κάνει ένα μόνο πέρασμα.

Στα στατικά δεδομένα υπάρχει πληθώρα αλγορίθμων για τον υπολογισμό συχνών στοιχείων. Οι αλγόριθμοι αυτοί έχουν βελτιωθεί σημαντικά, δίνοντας ακριβή διαστήματα έπειτα από περιορισμένο αριθμό σαρώσεων. Οι καλύτεροι εξ αυτών απαιτούν μόνο δύο σαρώσεις. Ωστόσο στα δεδομένα συνεχούς ροής, που μπορεί να γίνει μόνο ένα πέρασμα και τα αποτελέσματα να δίνονται σε σύντομο χρονικό διάστημα, η απόδοση των αλγορίθμων αυτών πέφτει (Jiang, Gruenwald, 2006) .

3.5.2 Αλγόριθμοι με συμπαγή δομή

Οι αλγόριθμοι εύρεσης συχνών στοιχείων με συμπαγή δομή εστιάζουν στην αποτελεσματική εξαγωγή των πιο συχνών μοτίβων δεδομένων, μειώνοντας τον αριθμό των υποψήφιων συχνών συνδυασμών, χωρίς να χάνεται η πληροφορία.

Ο παρακάτω αλγόριθμος χρησιμοποιεί μία δομή πλέγματος για την αποθήκευση των δεδομένων, την κατά προσέγγιση συχνότητα τους και τα πιθανά σφάλματα στην κατά προσέγγιση συχνότητα. Ο αλγόριθμος δέχεται ως είσοδο δεδομένα από ένα σύνολο I και δύο παραμέτρους που ορίζονται από το χρήστη: ένα όριο υποστήριξης $s \in (0,1)$ και ένα σφάλμα παραμέτρου $\varepsilon \in (0,1)$

Έστω S το μήκος της ροής (μπορεί να είναι πλήθος συνόλων ή πλήθος μεμονωμένων στοιχείων). Όλα τα στοιχεία των οποίων η πραγματική συχνότητα ξεπερνάει το s θα αποτελούν output. Θα πρέπει να σημειωθεί πως οι εκτιμώμενες συχνότητες είναι μεγαλύτερες από τις κανονικές.

Η δομή δεδομένων είναι ένα σύνολο καταχωρήσεων της μορφής $(set, f, \Delta) \in D$, όπου set είναι ένα υποσύνολο δεδομένων, f είναι ένας ακέραιος που αντιπροσωπεύει την κατά προσέγγιση συχνότητα και Δ είναι το μέγιστο δυνατό σφάλμα στο f. Τα δεδομένα που εισέρχονται διαμερίζονται σε κάδους, όπου κάθε κάδος αποτελείται από $w=1/\varepsilon$ στοιχεία. Κάθε κάδος έχει μία ετικέτα (bucket id), ξεκινώντας από το 1. Ο τρέχων κάδος δηλώνεται με $b_{current}$.

Είναι σημαντικό να τονιστεί πως δεν γίνεται επεξεργασία των δεδομένων ανά στοιχείο, αλλά γίνεται προσπάθεια να γεμίσει η διαθέσιμη μνήμη με όσον το δυνατόν περισσότερα δεδομένα και στη συνέχεια αυτά να επεξεργαστούν σε παρτίδες (batches). Έστω β ο αριθμός των κάδων που υπάρχουν στην μνήμη όταν η τρέχουσα παρτίδα υποβάλλεται σε επεξεργασία. Είναι σημαντικό το β να είναι μεγάλος αριθμός, καθώς οποιοδήποτε υποσύνολο του I εμφανίζεται περισσότερες από $\beta+1$ φορές συμβάλει ως είσοδος στο D. Το D ανανεώνεται ως εξής.

- Για κάθε είσοδο$(set, f, \Delta) \in D$ το f ανανεώνεται μετρώντας τις εμφανίσεις του set στην τρέχουσα δεσμίδα. Αν η ενημερωμένη καταχώρηση ικανοποιεί τη σχέση $f+\Delta < b_{current}$, τότε η καταχώρηση διαγράφεται. (UPDATE_SET)

- Αν ένα set έχει συχνότητα $f \geq \beta$ στην τρέχουσα δεσμίδα και το set δεν εμφανίζεται στο D, δημιουργείται μία νέα είσοδος ($set, f, b_{current}$ -β). (NEW_SET)

Κάθε *set* του οποίου η πραγματική συχνότητα *fset*>ε, έχει μία καταχώρηση στο *D*. Επίσης αν μία καταχώρηση $(set, f, \Delta) \in D$, τότε η πραγματική συχνότητα *fset* ικανοποιεί την ανισότητα $f \leq fset \leq f + \Delta$. Όταν ο χρήστης ζητάει μία λίστα από αντικείμενα με *threshold*≤ ε, οι καταχωρήσεις εξάγονται στο *D*.

Η υλοποίηση έχει τρείς ενότητες: BUFFER, TRIE και SETGEN. Η πρώτη ενότητα (BUFFER) διαβάζει μια παρτίδα συναλλαγών στη διαθέσιμη κύρια μνήμη. Η δεύτερη ενότητα (TRIE) διατηρεί τη δομή δεδομένων που περιεγράφηκε πιο πάνω και η τρίτη ενότητα (SETGEN) εκτελείται στην τρέχουσα παρτίδα και απαριθμεί τα σύνολα των δεδομένων, μαζί με τις συχνότητές τους. Οι δύο τελευταίες ενότητες υλοποιούν τα UPDATE_SET και NEW_SET. Η πρόκληση έγκειται στο σχεδιασμό ενός χώρου με αποδοτική αναπαράσταση της δομής δεδομένων που θα διατηρείται από το (TRIE), η οποία χρησιμοποιεί ένα γρήγορα αλγόριθμο για την καταμέτρηση των στοιχείων και των συχνοτήτων εμφάνισής τους. Παρακάτω περιγράφονται αναλυτικά οι τρείς ενότητες

Το BUFFER αποτελεί μία λειτουργική μονάδα η οποία διαβάζει τα δεδομένα από μια παρτίδα στην κύρια μνήμη. Κάθε δεδομένο έχει έναν αναγνωριστικό κωδικό (id) και είναι τοποθετημένο το ένα μετά το άλλο. Σε μία παρτίδα, το BUFFER ταξινομεί κάθε δεδομένο με βάση το αναγνωριστικό του στοιχείο.

Το TRIE αποτελεί μία λειτουργική μονάδα η οποία διατηρεί τη δομή δεδομένων *D*. Εννοιολογικά είναι ένα δάσος, δηλαδή ένα σύνολο από δέντρα, που αποτελείται από επισημασμένους κόμβους των οποίων οι ετικέτες έχουν τη μορφή (*item_id, f, Δ, level*). Με *item_id* συμβολίζεται ο αναγνωριστικός κωδικός του κάθε δεδομένου με *f* η εκτιμόμενη συχνότητα, με *Δ* το μέγιστο πιθανό σφάλμα και *level* είναι η απόσταση του κόμβου από την ρίζα του δέντρου στο οποίο ανήκει. Το *level* της ρίζας είναι 0. Το *level* του κάθε κόμβου είναι ένα παραπάνω από το *level* των γονικών του κόμβων. Τα παιδιά οποιουδήποτε κόμβου ταξινομούνται από το *item_id* τους, όπως και οι ρίζες όλων των δέντρων του δάσους. Ο κόμβος ενός δέντρου αντιπροσωπεύει ένα σύνολο στοιχείων που αποτελείται από *item_ids*, καθώς και

όλους τους προγόνους του. Υπάρχει μία ένα προς ένα αντιστοιχία μεταξύ των εισόδων στο *D* και των κόμβων στο TRIE.

Το SETGEN αποτελεί μία λειτουργική μονάδα που δημιουργεί υποσύνολα από *item_id* μαζί με τις συχνότητές τους στην τρέχουσα, υπό επεξεργασία παρτίδα. Παρατηρώντας την περιγραφή των λειτουργιών UPDATE SET και NEW SET γίνεται εύκολα αντιληπτό ότι ένα υποσύνολο πρέπει να απαριθμείται, είτε εμφανίζεται στο TRIE είτε η συχνότητά του στην τρέχουσα παρτίδα υπερβαίνει το β.

Συνοψίζοντας η διαδικασία εξελίσσεται με πρώτο το BUFFER να γεμίζει τη διαθέσιμη μνήμη με όσον το δυνατόν περισσότερα δεδομένα, τα οποία και ταξινομεί. Στη συνέχεια το TRIE εκτελείται στην τρέχουσα παρτίδα, δημιουργώντας σύνολα από αντικείμενα μαζί με τις εκτιμώμενες συχνότητές τους και στη συνέχεια μειώνει το πλήθος των υποσυνόλων χρησιμοποιώντας την τεχνική κλαδέματος (pruning). Μαζί τα TRIE και SETGEN υλοποιούν τα UPDATE SET και NEW SET. Στο τέλος το TRIE αποθηκεύει την ανανεωμένη δομή δεδομένων και το BUFFER ετοιμάζεται να διαβάσει την επόμενη παρτίδα

Παρακάτω δίνεται πιο συνοπτικά η δομή του αλγορίθμου, ο οποίος έχει τις ακόλουθες παραμέτρους.

s: Όριο υποστήριξης που ορίζεται από το χρήστη ($0 < s < 1$).

ε: Παράμετρος σφάλματος που ορίζεται από το χρήστη ($0 < \varepsilon < 1$).

w: Μέγεθος κάδου, ορίζεται ως $1/\varepsilon$.

β: Αριθμός κάδων στη μνήμη κατά την επεξεργασία μιας παρτίδας.

Βήμα 1: Ανάγνωση δεδομένων (BUFFER):

Το BUFFER γεμίζει τη διαθέσιμη μνήμη με δεδομένα από μια παρτίδα συναλλαγών και τα ταξινομεί με βάση έναν αναγνωριστικό κωδικό (item_id), που διαθέτει καθένα από αυτά

Βήμα 2 : Επεξεργασία των δεδομένων (TRIE & SETGEN):

Το TRIE διατηρεί μια δομή δεδομένων D που αντιστοιχεί σε ένα δάσος από δέντρα. Κάθε κόμβος του TRIE αντιπροσωπεύει ένα σύνολο στοιχείων και περιέχει την εκτιμώμενη συχνότητα f και το μέγιστο δυνατό σφάλμα Δ. Η συχνότητα f ενημερώνεται μετρώντας τις εμφανίσεις του συνόλου στοιχείων στην τρέχουσα παρτίδα (batch). Αν η καταχώρηση ικανοποιεί τη σχέση $f+\Delta < b_{current}$, τότε η καταχώρηση διαγράφεται (UPDATE_SET), ενώ αν εμφανίζεται με συχνότητα $f \geq \beta$ (όπου β είναι ο αριθμός των κάδων που υπάρχουν στη μνήμη) και δεν υπάρχει ήδη στο TRIE, δημιουργείται μία νέα καταχώρηση με τη μορφή $(set, f, b_{current} - \beta)$ (NEW_SET).

Το SETGEN δημιουργεί υποσύνολα στοιχείων (set) μαζί με τις συχνότητες εμφάνισής τους στην τρέχουσα παρτίδα. Απαριθμεί τα υποσύνολα δεδομένων που είτε υπάρχουν ήδη στο TRIE είτε εμφανίζονται στην παρτίδα με συχνότητα μεγαλύτερη από β.

Βήμα 3: Ανανέωση της δομής δεδομένων (TRIE):

Η δομή TRIE αποθηκεύει την ανανεωμένη κατάσταση της δομής δεδομένων D, μετά την επεξεργασία της τρέχουσας παρτίδας. Στη συνέχεια εφαρμόζει κλάδεμα (pruning) για να μειώσει το πλήθος των συνόλων που διατηρεί, διαγράφοντας όσα υποσύνολα δεν ικανοποιούν τις προϋποθέσεις συχνότητας.

Βήμα 4: Εκτέλεση του αλγορίθμου σε επόμενες παρτίδες:

Το BUFFER ετοιμάζεται να διαβάσει την επόμενη παρτίδα δεδομένων, επαναλαμβάνοντας τη διαδικασία για τα νέα δεδομένα (Manku, Motwani, 2002).

ΚΕΦΑΛΑΙΟ 4

Αποτελέσματα και Ερμηνεία

4.1 Περιγραφή των δεδομένων

Τα δεδομένα που θα αναλυθούν αφορούν σε ασθενείς covid που παρακολουθήθηκαν με έξυπνα ρολόγια (smart watches). Θα πρέπει να σημειωθεί ότι στα δεδομένα αυτά έχουν εφαρμοστεί τεχνικές ανωνυμοποίησης, ώστε να μην μπορούν με κανένα τρόπο να συσχετιστούν με τους πραγματικούς χρήστες.

Οι μετρήσεις περιλαμβάνουν:

- Καθημερινή Δραστηριότητα – βήματα, δηλαδή τα βήματα, την απόσταση σε χιλιόμετρα που διανύθηκε και τις θερμίδες που καταναλώθηκαν από τους ασθενείς.
- Καθημερινή Δραστηριότητα – ύπνος, δηλαδή τα λεπτά συνολικού ύπνου, βαθύ, ελαφρύ και REM, καθώς επίσης και τα λεπτά που ήταν ξύπνιοι οι ασθενείς.
- Μετρήσεις βιοσημάτων (καρδιακοί παλμοί, συστολική πίεση, διαστολική πίεση, κιλά, αναπνευστικός ρυθμός, SPO2).
- Απαντήσεις σε καθημερινά ερωτηματολόγια που αφορούν σε βαθμό δύσπνοιας και καταγραφή συμπτωμάτων.

Τα δεδομένα στην καθημερινή δραστηριότητα ύπνου και βημάτων, στην πλειοψηφία τους, προέρχονται από γυναίκες. Συγκεκριμένα, καταγράφονται δεδομένα από οχτώ γυναίκες και μόνο δύο άνδρες. Ωστόσο στις μετρήσεις βιοσημάτων υπάρχει ισορροπία μεταξύ των δύο φύλων. Παράλληλα στα δεδομένα αυτά παρατηρούνται ελλιπείς καταγραφές οι οποίες αποδίδονται σε άτομα που εγκατέλειψαν την έρευνα ή προσπέρασαν την καταγραφή άθελά τους. Τα ελλιπή αυτά δεδομένα θεωρούνται λοιπόν MCAR, δηλαδή εντελώς τυχαία ελλιπή δεδομένα και δεν θεωρείται ότι επηρεάζουν τα τελικά αποτελέσματα. Στο τελευταίο σύνολο δεδομένων, που αφορά σε απαντήσεις σε ερωτηματολόγιο, δημιουργείται ένα σκορ το οποίο

αθροίζει το βαθμό δύσπνοιας με το πλήθος των συμπτωμάτων. Πιο συγκεκριμένα δίνονται οι ακόλουθες απαντήσεις για το βαθμό δύσπνοιας:

1.Καθόλου δύσπνοια ή δύσπνοια μόνο σε κοπιαστική εργασία

2.Εμφάνιση δύσπνοιας όταν περπατάς γρήγορα σε επίπεδο έδαφος ή σε μικρή κλίση

3.Περπατάς πιο αργά από άτομα της ίδιας ηλικίας σε επίπεδο έδαφος λόγω δύσπνοιας ή σταματάς για ανάσα αν περπατάς μόνος σου

4. Εμφάνιση δύσπνοιας όταν περπατάς μερικά μέτρα σε επίπεδο έδαφος.

5.Παραμένεις στο σπίτι λόγω της δύσπνοιας και δυσκολεύεσαι ακόμα και να ντυθείς

Δεν παρατηρείται καμία απάντηση για το 4.

Όσον αφορά στα συμπτώματα δίνονται οι παρακάτω απαντήσεις:

Κόπωση, αδυναμία, αγευσία, ανοσμία, ρινική συμφόρηση, πυρετός, κεφαλαλγία, αρθραλγία, βήχας, πονόλαιμος, φτέρνισμα, μυαλγία, ανορεξία, αϋπνία, καταρροή, ναυτία, δεκατικός πυρετός, δερματικό εξάνθημα, δύσπνοια, κοιλιακό άλγος-διάρροια, νυχτερινή εφίδρωση, ταχυκαρδίες, στερνικό άλγος, υπνηλία, ρίγη, αλλοίωση στη χροιά της φωνής ,Άλλο

Το σκορ προκύπτει αθροίζοντας το πλήθος των συμπτωμάτων με τον βαθμό δύσπνοιας. Στο dataset υπάρχουν ορισμένες εγγραφές στις οποίες έχει καταγραφεί πολλές φορές το ίδιο σύμπτωμα. Αυτές θεωρούνται σφάλματα, επομένως το κάθε σύμπτωμα προσμετράται μόνο μία φορά. Οι ασθενείς εγκαταλείπουν την έρευνα μόλις τα συμπτώματά τους υποχωρήσουν, για το λόγο αυτό δεν υπάρχει το ίδιο πλήθος δεδομένων για όλους τους ασθενείς. Αρχικά τα δεδομένα προετοιμάζονται για την επεξεργασία τους. Θα ανιχνευτούν ελλιπή, ασυνεπή δεδομένα αλλά και ακραίες τιμές μέσω της οπτικοποίησης και κάνοντας χρήση του αλγορίθμου Local Outlier Factor. Στη συνέχεια για την ανάλυσή τους θα χρησιμοποιηθούν μέθοδοι σύνοψης και συγκεκριμένα τα Ιστογράμματα. Τέλος θα εφαρμοστούν συσταδοποίηση και εύρεση συχνών μοτίβων. Παρακάτω δίνεται ένας συγκεντρωτικός πίνακας των μεταβλητών μαζί με άλλα στατιστικά περιγραφικά στοιχεία για την καθεμία.

ΜΕΤΑΒΛΗΤΗ	ΜΟΝΑΔΑ	ΣΥΝΟΛΟ	ΜΕΣΗ ΤΙΜΗ	ΔΙΑΣΠΟΡΑ	MIN	MAX
Steps	N	260	8579.5	31468277.78	17	1730529
Calories	N	260	170.38	14889.76	0	683
Distance	m	260	5913.87	14252630.08	0	19805
Awake	min	238	3.63	95.31	0	81
Total sleep	min	238	458.04	6087.22	150	716
Deep sleep	min	238	126.01	1993.14	31	223
Rem sleep	min	238	100.46	1022.77	18	194
Light sleep	min	238	231.57	2199.72	84	360
Diastolic Pressure	mmHg	2216	125.55	71.17	100.0	140.0
Systolic Pressure	mmHg	2216	69.98	79.78	45.0	125.0
Heartrate	bpm	133647	77.23	145.92	37.0	180.0
Weight	kg	2205	71.16	224.56	16.0	135.0
Respiratory Rate	br/min	2202	15.36	3.83	11.0	22.0
Body Temperature	°C	2238	36.79	0.34	35.9	40.0
SPO2	%	12282	96.94	5.98	71.0	100.0
Questionnaire score	N	2010	2.0	3.87	1.0	18.0

Πίνακας 4-1: Συγκεντρωτικός πίνακας των μεταβλητών με στατιστικά περιγραφικά στοιχεία

4.2 Οπτικοποίηση

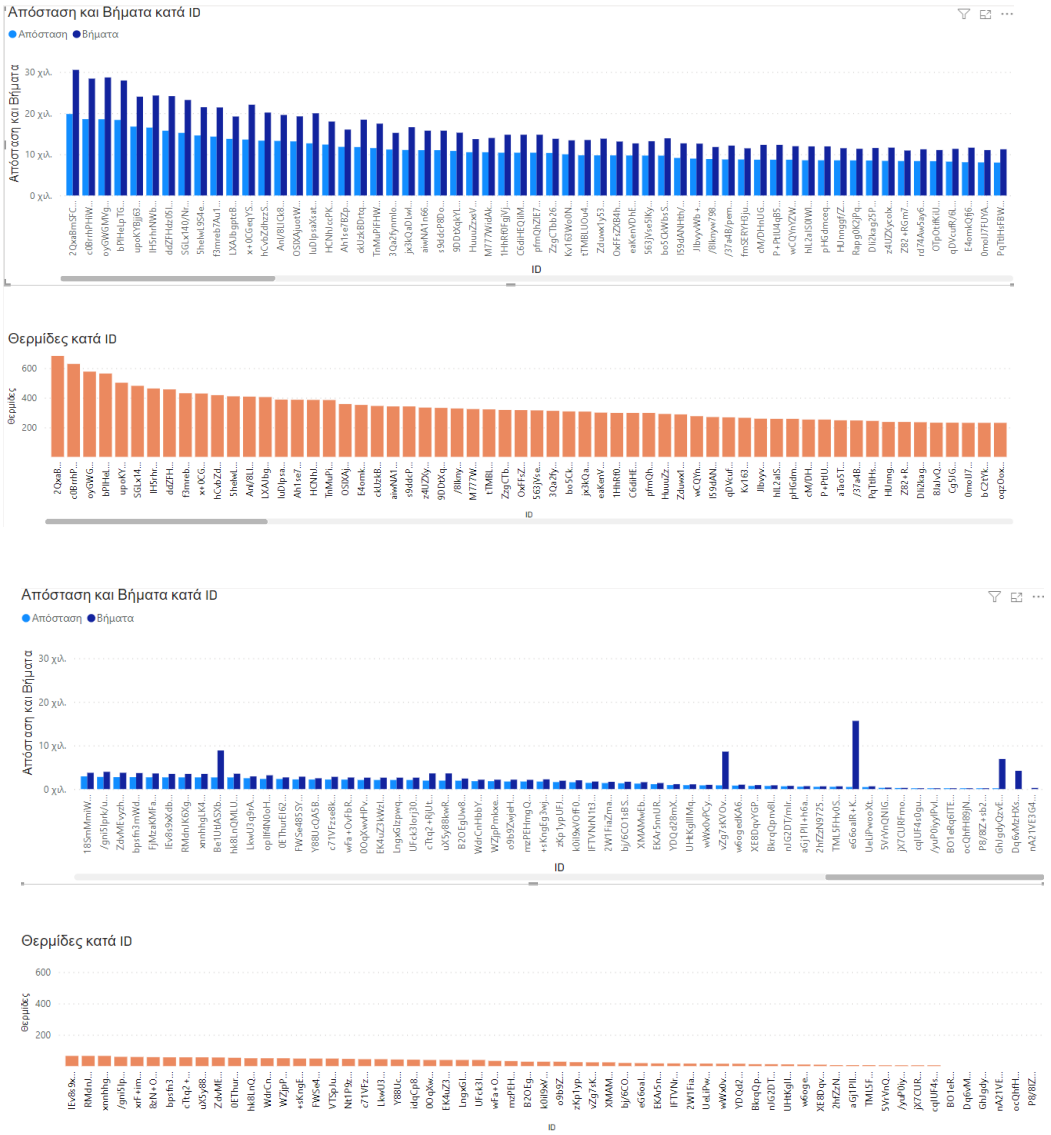
Στο κεφάλαιο αυτό θα χρησιμοποιηθούν bar charts και column charts μέσω του Power BI προκειμένου να οπτικοποιηθούν τα δεδομένα και να αποκτηθεί μια αρχική εικόνα του εύρους των τιμών τους. Συγχρόνως θα εντοπισθούν ακραίες και ελλιπείς τιμές.

Οι πρώτες μετρήσεις αφορούν σε καθημερινή δραστηριότητα ύπνου και ειδικότερα καταγράφονται τα λεπτά συνολικού ύπνου, τα οποία χωρίζονται σε τρείς κατηγορίες, δηλαδή σε βαθύ ύπνο, ελαφρύ ύπνο και ύπνο rem. Στα παρακάτω stacked bar charts απεικονίζονται τρία στιγμιότυπα από συσσωρευτικές μετρήσεις ελαφρύ ύπνου, βαθύ ύπνου και ύπνου rem που τοποθετημένες η μία δίπλα στην άλλη απεικονίζουν και το συνολικό ύπνο. Όπως έχει αναφερθεί και πιο πάνω τα δεδομένα είναι συνθετικά, συνεπώς στον κατακόρυφο άξονα απεικονίζεται ο κωδικός id με τον οποίο έχουν παραχθεί.

Σχήμα 4-1:Γράφημα συσσωρευμένων ράβδων σε Power Bi για τα δεδομένα ύπνου

Όπως φαίνεται παραπάνω υπάρχουν κάποιες αρκετά μικρές και κάποιες αρκετά μεγάλες τιμές. Πιο συγκεκριμένα υπάρχει καταγραφή για 150 λεπτά συνολικού ύπνου, δηλαδή 2.5 ώρες και καταγραφή για 716 λεπτά ύπνου δηλαδή σχεδόν 12 ώρες. Και οι δύο αυτές τιμές αποκλίνουν σημαντικά από τις υπόλοιπες και για το λόγο αυτό θεωρούνται ακραίες τιμές.

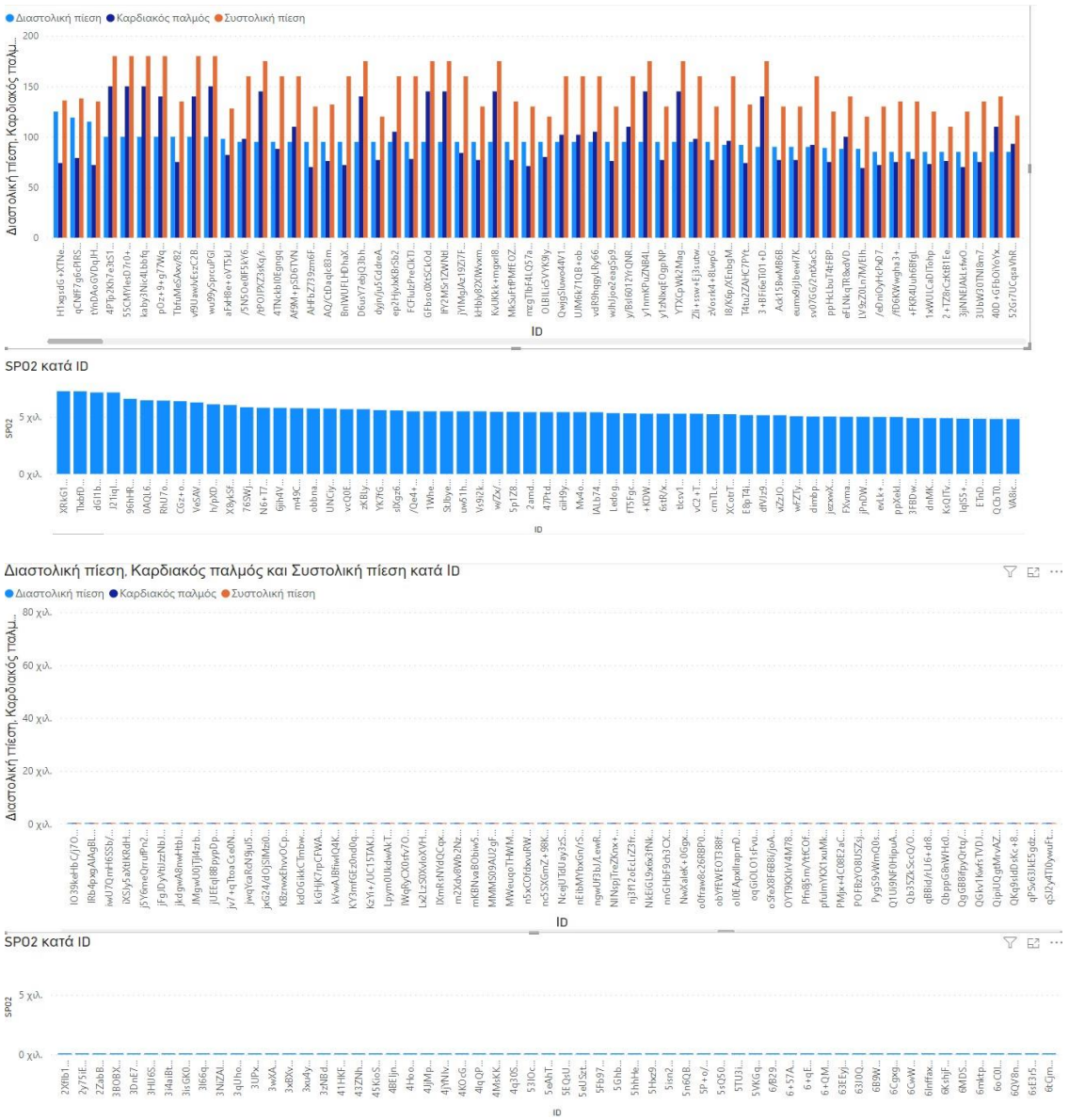
Οι επόμενες μετρήσεις αφορούν σε καθημερινή δραστηριότητα βημάτων. Ειδικότερα στα δύο στιγμιότυπα με τα μπλε column charts καταγράφονται τα βήματα και τα χιλιόμετρα που διανύονται και στα δύο στιγμιότυπα με τα πορτοκαλί column charts οι θερμίδες που καταναλώνονται.



Σχήμα 4-2:Γραφήματα ομαδοποιημένων στηλών σε Powe Bi για τα δεδομένα βημάτων

Παρατηρείται μεγάλο έυρος τιμών χωρίς κάποια σημαντική απόκλιση. Δεν υπάρχει δηλαδή κάποια τιμή που να διαφέρει σημαντικά από τις υπόλοιπες. Ωστόσο φαίνεται να υπάρχουν καταγραφές με ελάχιστη τιμή στις μετρήσεις. Πιο συγκεκριμένα στα δυο τελευταία στιγμιότυπα εντοπίζονται καταγραφές με ελάχιστες, ακόμα και μηδενικές θερμίδες, βήματα και χιλιόμετρα, κάτι που σημαίνει ότι οι χρήστες δεν περπάτησαν καθόλου στο διαστημα μίας μέρας.

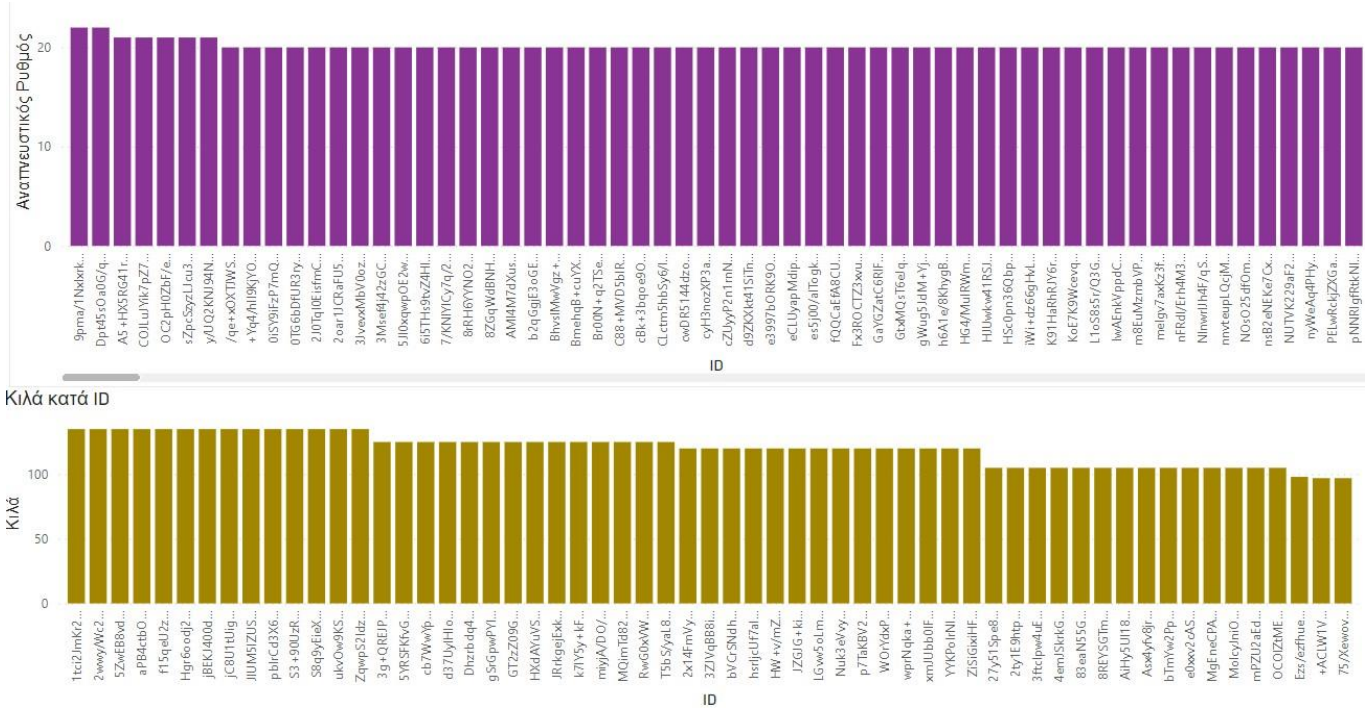
Οι επόμενες μετρήσεις αφορούν σε μετρήσεις βιοσημάτων, δηλαδή στη συστολική και διαστολική πίεση και στον καρδιακό παλμό στο πρώτο column chart καθώς και στα επίπεδα SPO2 στο μπλε column chart.



Σχήμα 4-3:Γραφήματα ομαδοποιημένων στηλών σε Powe Bi για τις βιομετρικές μετρήσεις πίεσης, SPO2 και καρδιακών παλμών

Από την παραπάνω απεικόνιση και συγκεκριμένα από τα δεύτερα στιγμιότυπα με τις μηδενικές τιμές φαίνεται να μην υπάρχουν καταγεγραμμένες τιμές για τις ποσότητες αυτές σε όλες τις εγγραφές, καθώς μηδενική τιμή καρδιακών παλμών πίεσης και SPO2 δεν υφίσταται. Εντοπίζονται δηλαδή ελλιπή δεδομένα.

Οι παρακάτω μετρήσεις αναφέρονται στον αναπνευστικό ρυθμό που απεικονίζεται στο μοβ column chart και τα κιλά που απεικονίζονται στο κίτρινο column chart. Τα δεδομένα δε φαίνεται να χρειάζονται κάποια διόρθωση.



Σχήμα 4-4:Γραφήματα στηλών σε Powe Bi για τις μετρήσεις αναπνευστικού ρυθμού και κιλών

Τέλος στο bar char που ακολουθεί απεικονίζονται οι καταγραφές απαντήσεων σε καθημερινά ερωτηματολόγια. Οι ερωτήσεις αφορούν στο βαθμό δύσπνοιας, η οποία βαθμολογείται με έναν αριθμό. Ζητείται επίσης από τους συμμετέχοντες να επιλέξουν μεταξύ ορισμένων συμπτωμάτων. Στο τέλος για κάθε καταγραφή δημιουργείται ένα σκορ το οποίο αθροίζει το βαθμό δύσπνοιας με το πλήθος των συμπτωμάτων, όπως αναφέρθηκε και παραπάνω.

Σχήμα 4-5:Γραφήματα ομαδοποιημένων στηλών σε Powe Bi για τα σκορ των απαντήσεων στα ερωτηματολόγια

Από την παραπάνω απεικόνιση φαίνεται πολλές καταγραφές να έχουν σκορ ένα, το οποίο ερμηνεύεται ως βαθμός δύσπνοιας 1 χωρίς συμπτώματα. Υπάρχουν ωστόσο και εγγραφές που συγκεντρώνουν μεγάλο σκορ. Οι δύο πρώτες έχουν 18 και 14 αντίστοιχα, αποκλίνοντας σημαντικά από τις υπόλοιπες. Οι εγγραφές αυτές θεωρούνται ακραίες τιμές.

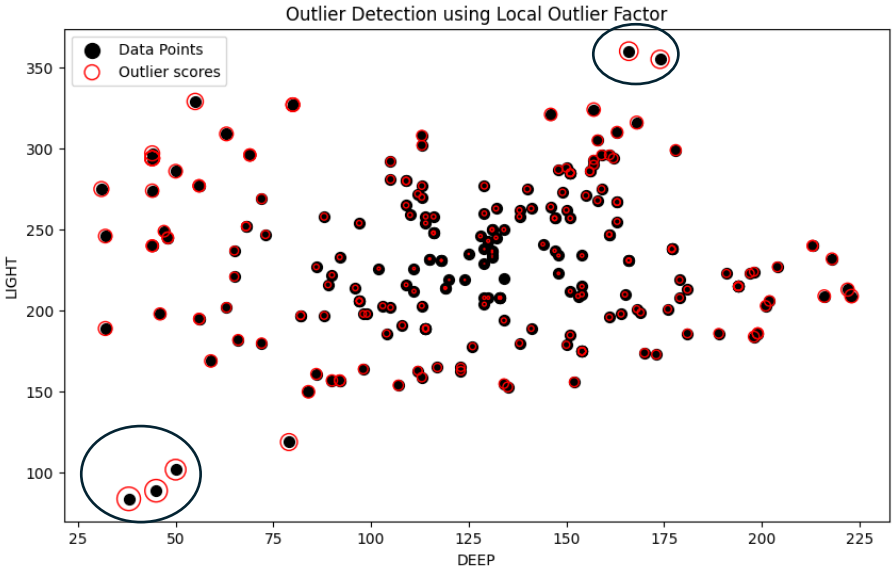
Μέσω της οπτικοποίησης των δεδομένων προκύπτουν τα εξής συμπεράσματα:

- Στα δεδομένα ύπνου παρατηρούνται ακραίες τιμές, τόσο μεγάλες όσο και μικρές, που αποκλίνουν σημαντικά από τις υπόλοιπες.
- Στα δεδομένα καθημερινής άσκησης παρατηρούνται κάποιες μηδενικές τιμές που ερμηνεύονται ως έλλειψη άσκησης του χρήστη, οι οποίες θεωρούνται φυσιολογικές.
- Στις μετρήσεις βιοσημάτων και συγκεκριμένα στην πίεση, τον αναπνευστικό ρυθμό και το SPO2 παρατηρούνται ελλιπείς τιμές, καθώς δεν υφίσταται οι ποσότητες αυτές να είναι μηδέν.
- Στα σκορ από τις απαντήσεις σε ερωτηματολόγια παρατηρούνται ορισμένες πολύ μεγάλες τιμές που αποκλίνουν σημαντικά από τις υπόλοιπες. Διαπιστώνεται κατόπιν διερεύνησης πως προκύπτουν από επαναληπτική καταγραφή των ίδιων συμπτωμάτων

4.3 Local Outlier Factor

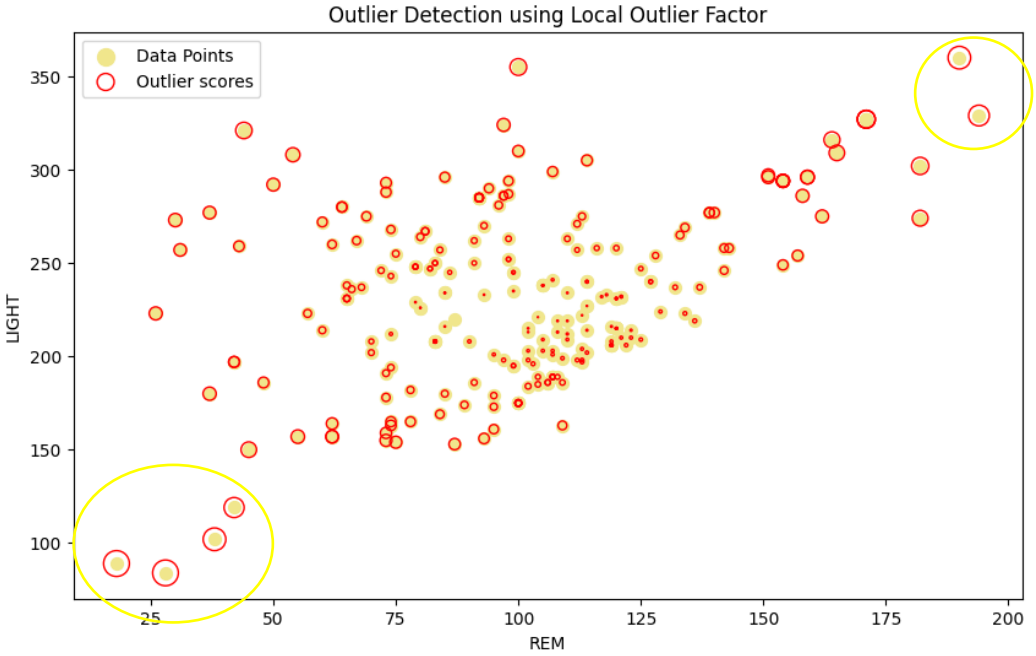
Στο κεφάλαιο αυτό θα γίνει χρήση του αλγορίθμου Local Outlier Factor με σκοπό την ανίχνευση ακραίων τιμών στο σύνολο των δεδομένων. Ο αλγόριθμος που χρησιμοποιείται καλεί το μοντέλο Local Outlier Factor που δέχεται ως μεταβλητή το πλήθος των γειτόνων και στη συνέχεια προσαρμόζει σε αυτό τα δεδομενα. Τέλος υπολογίζεται το LOF. Όταν το LOF ξεπερνάει σημαντικά το 1, τότε το αντίστοιχο δεδομένο ενδέχεται να είναι ακραία τιμή (outlier). Το μέγεθος του LOF είναι ανάλογο με την ακτίνα του κόκκινου κυκλικού δίσκου. Συνεπώς τα στοιχεία με μεγάλη ακτίνα είναι outliers. Για όλα τα σύνολα δεδομένων δίνεται πλήθος γειτόνων 100.

Όσον αφορά στα δεδομένα ύπνου αρχικά συγκρίνονται τα ποσοστά ελαφρού ύπνου και βαθιού ύπνου. Προκύπτει ότι τα περισσότερα δεδομένα είναι συγκεντρωμένα στο κέντρο του γραφήματος. Σημαντικά outliers εντοπίζονται στην κάτω αριστερή γωνία με τιμές βαθιού ύπνου στο διάστημα (25,50) και ελαφρού ύπνου στο διάστημα (0,100). Επιπλέον outliers εντοπίζονται στο πάνω δεξιό μέρος του γραφήματος με τιμές ελαφρού ύπνου στο διάστημα (350,400) και βαθιού ύπνου στο διάστημα (150,175). Οι πρώτες τιμές πιθανόν αφορούν σε χρήστες με σχετικά μικρή διάρκεια ύπνου και οι δεύτερες με σχετικά μεγάλη διάρκεια.



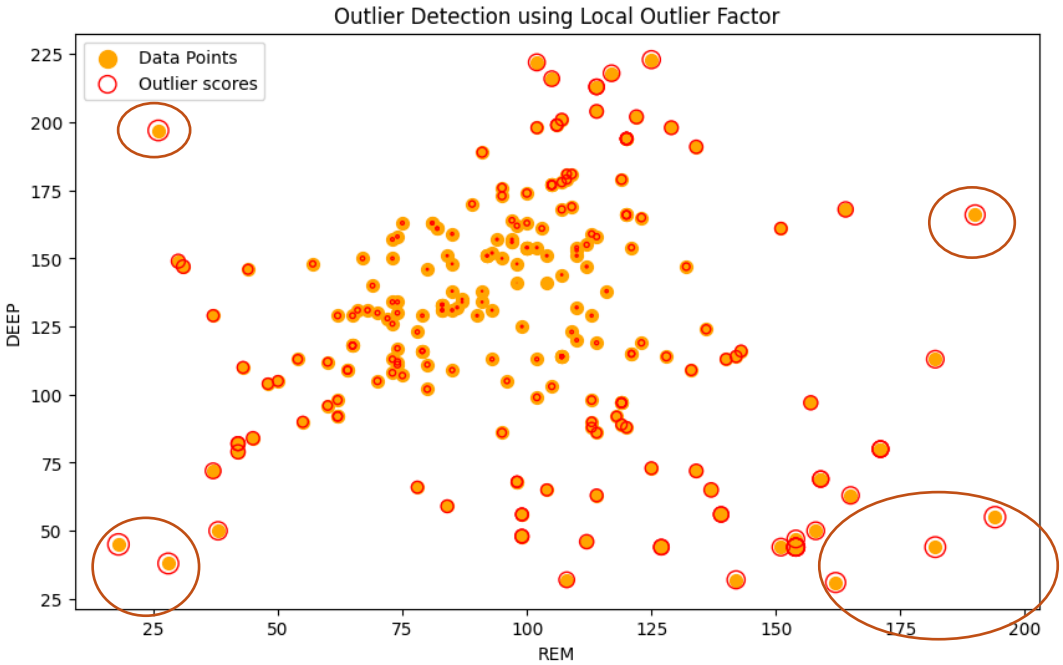
Σχήμα 4-6: Local Outlier Factor για ελαφρύ ύπνο και βαθύ ύπνο

Στη συνέχεια συγκρίνονται τα ποσοστά ελαφρού ύπνου και ύπνου rem. Και εδώ τα περισσότερα δεδομένα είναι συγκεντρωμένα στο κέντρο του γραφήματος και παρατηρούνται σημαντικά outliers στην κάτω αριστερή γωνία με ύπνο rem στο διάστημα (0,50) και ελαφρύ ύπνο στο διάστημα (0,125). Παρατηρούνται επίσης outliers στην πάνω δεξιά γωνία με ύπνο rem στο διάστημα (175,200) και ελαφρύ ύπνο στο διάστημα (300,350).



Σχήμα 4-7: Local Outlier Factor για ελαφρύ ύπνο και ύπνο rem

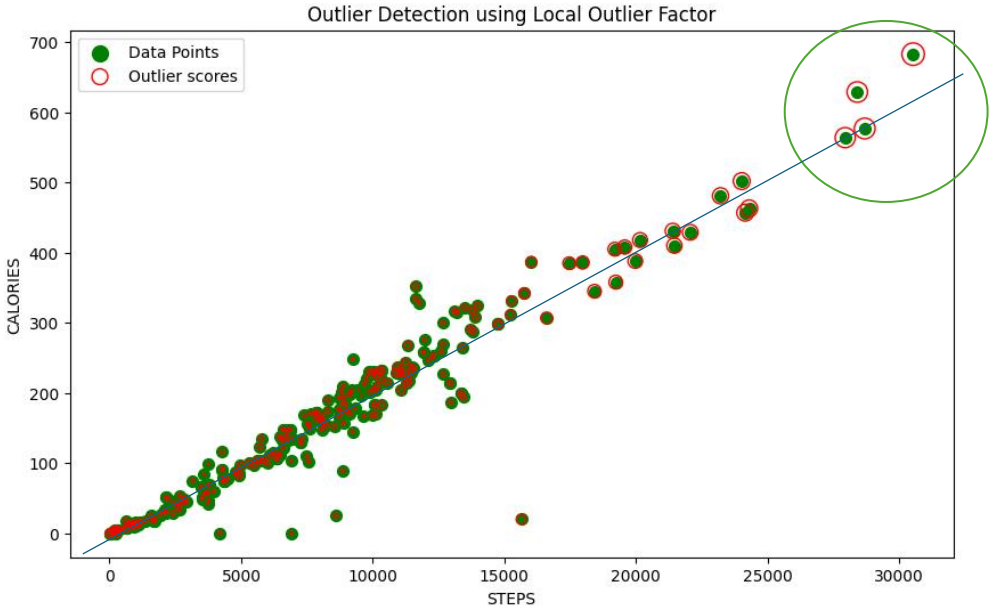
Τέλος συγκρίνονται τα ποσοστά ύπνου rem και βαθιού ύπνου. Όπως και στα υπόλοιπα γραφήματα το μεγαλύτερο μέρος των σημείων συγκεντρώνεται στη μέση. Παρατηρούνται σημαντικά outliers στην κάτω αριστερή γωνία με ύπνο rem στο διάστημα (0,25) και βαθύ ύπνο στο διάστημα (25,50). Παρατηρούνται επίσης outliers στην κάτω δεξιά γωνία με ύπνο rem στο διάστημα (150,200) και βαθύ ύπνο στο διάστημα (25,50). Τέλος υπάρχουν δύο ακραίες τιμές μία πάνω αριστερά με βαθύ ύπνο κοντά στο 200 και ύπνο rem κοντά στο 25 και μία πάνω δεξιά με βαθύ ύπνο στο διάστημα (150,175) και ύπνο rem στο διάστημα (175,200). Στην πρώτη παρατηρείται μικρή τιμή για τον ύπνο rem συγκριτικά με το βαθύ και στη δεύτερη ο ύπνος rem ξεπερνάει τον βαθύ ύπνο.



Σχήμα 4-8: Local Outlier Factor για ύπνο rem και βαθύ ύπνο

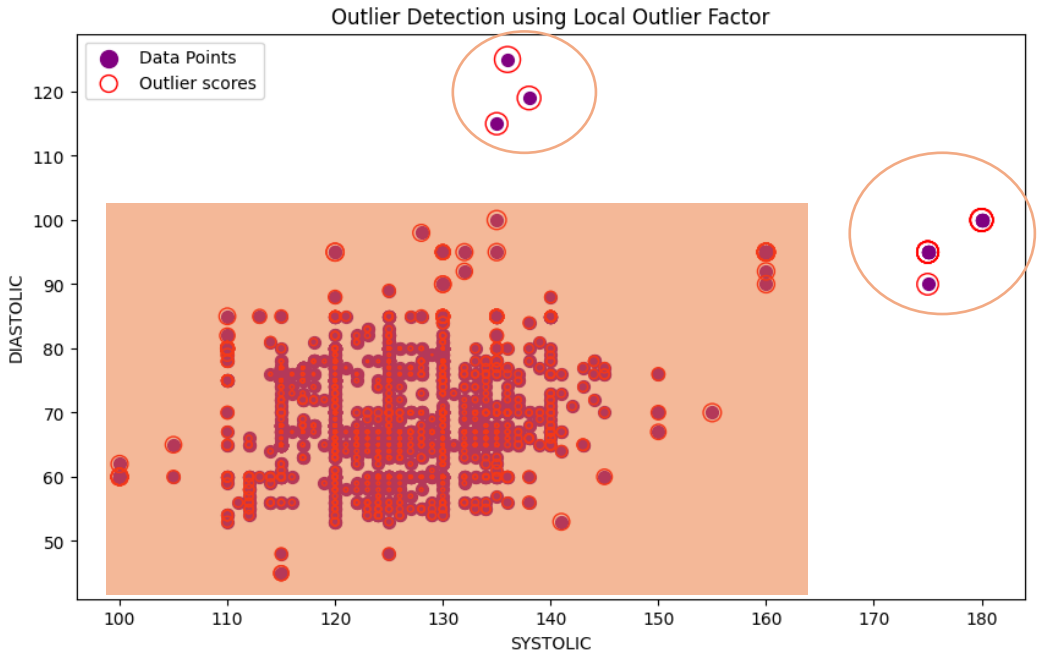
Για τα δεδομένα ύπνου, γίνεται εύκολα αντιληπτό πως και στα τρία γραφήματα, τα περισσότερα σημεία συγκεντρώνονται σε ένα κεντρικό τμήμα όπου δεν ανιχνεύονται outliers. Ωστόσο το καθένα από αυτά αναδεικνύει διαφορετικά outliers βάσει των χαρακτηριστικών που εξετάζονται.

Στη συνέχεια, στα δεδομένα που αφορούν στα βήματα και τις θερμίδες, εφαρμόζοντας τον αλγόριθμο προκύπτουν τα παρακάτω αποτελέσματα. Τα σημεία βρίσκονται, πλην ορισμένων εξαιρέσεων, σε μια νοητή ευθεία, κάτι αναμενόμενο, καθώς οι θερμίδες είναι ανάλογες με τα βήματα. Τα περισσότερα σημεία είναι συγκεντρωμένα στο κάτω αριστερό μέρος της ευθείας. Ως εκ τούτου τα τέσσερα σημεία που βρίσκονται στο πάνω δεξί μέρος της ευθείας θεωρούνται outliers.



Σχήμα 4-9: Local Outlier Factor για θερμίδες και βήματα

Όσον αφορά στα δεδομένα που προκύπτουν από μετρήσεις διαστολικής και συστολικής πίεσης, κατόπιν εφαρμογής του αλγορίθμου Local Outlier Factor, προκύπτουν τα ακόλουθα δεδομένα. Το μεγαλύτερο ποσοστό των μετρήσεων έχουν διαστολική πίεση 50 με 100 μονάδες και συστολική 100 με 150 μονάδες που είναι κοντά στις επιθυμητές τιμές. Ωστόσο υπάρχουν μετρήσεις που σημειώνουν μεγάλες τιμές διαστολικής και συστολικής πίεσης, οι οποίες ξεπερνούν κατά πολύ τα επιτρεπτά όρια. Για παράδειγμα στο δείγμα υπάρχει καταγεγραμμένη τιμή για διαστολική 100 και συστολική 180 και τιμή με διαστολική 130 και συστολική 140, ενώ τα επιτρεπτά όρια είναι 80 και 120. Οι τιμές αυτές θεωρούνται outliers.



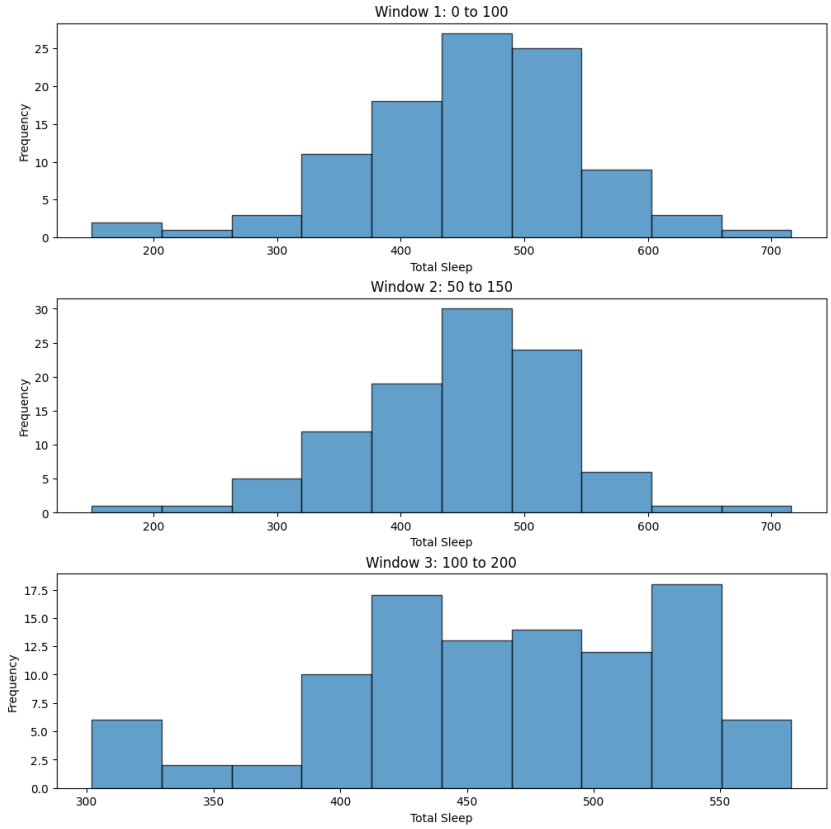
Σχήμα 4-10: Local Outlier Factor για διαστολική και συστολική πίεση

4.4 Ιστογράμματα Παραθύρου και Ιστογράμματα Εξασθένισης

Στο κεφάλαιο αυτό θα δημιουργηθούν ιστογράμματα παραθύρου και ιστογράμματα εξασθένισης για τα δεδομένα μετρήσεων και στη συνέχεια θα γίνει σύγκριση μεταξύ τους. Υπενθυμίζεται ότι τα ιστογράμματα παραθύρου περιλαμβάνουν τα τελευταία n δεδομένα, όπου n ένας προκαθορισμένος αριθμός, διαγράφοντας τα παλιά δεδομένα. Αντίστοιχα στα ιστογράμματα εξασθένισης, εξασθενίζεται εκθετικά η συνεισφορά των δεδομένων με την πάροδο του χρόνο, μέχρι να γίνει αμελητέα. Αξίζει να τονιστεί πως το ιστόγραμμα εξασθένισης μας δίνει μία εικόνα της συνολικής κατανομής των παρατηρήσεων με έμφαση στα τελευταία δεδομένα, καθώς τα πρώτα σταδιακά εξασθενούν. Τα ιστογράμματα παραθύρου από την άλλη δίνουν μια εικόνα της κατανομής των δεδομένων μόνο στο εκάστοτε παράθυρο. Συνεπώς τα δεύτερα αποτελούν έναν εύκολο τρόπο να παρατηρηθεί η διακύμανση στις τιμές των παρατηρήσεων στα διάφορα τμήματα της ροής. Στα δεδομένα της παρούσας διπλωματικής, για τα ιστογράμματα παραθύρου, το μέγεθος του παραθύρου ορίζεται όσο το πλήθος των

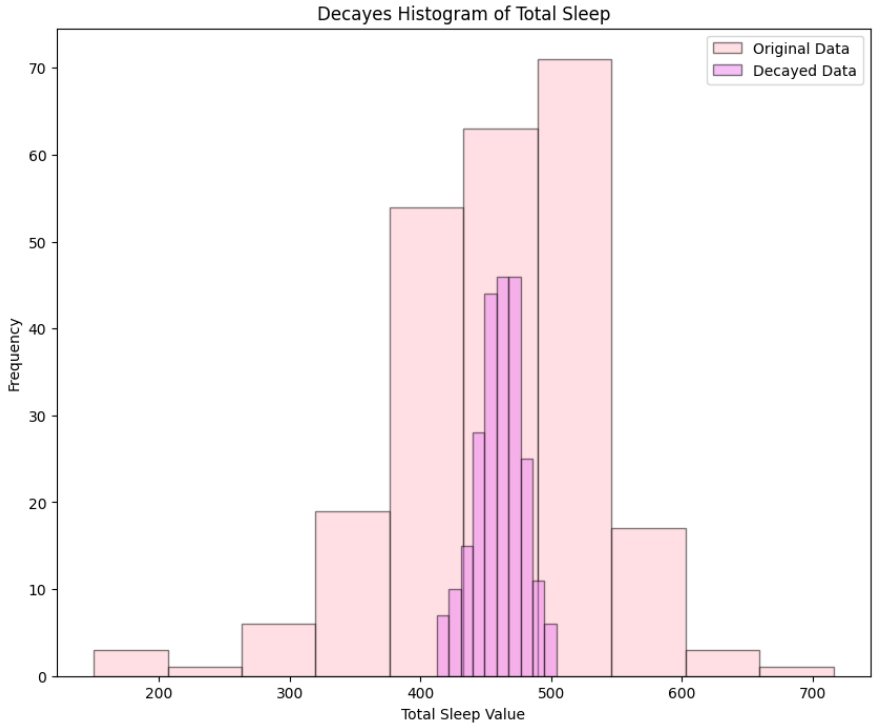
δεδομένων. Για τα ιστογράμματα εξασθένισης, ο πολλαπλασιαστής, ο οποίος και καθορίζει τον ρυθμό εξασθένισης είναι 0.1.

Παρακάτω δίνονται τα ιστογράμματα για τον συνολικό ύπνο. Στα δύο πρώτα ιστογράμματα παραθύρου η κατανομή έχει μία κορυφή κοντά στο διάστημα (450,500), είναι σχετικά συμμετρική και προσεγγίζει την κανονική κατανομή. Στο τρίτο ιστόγραμμα παραθύρου ωστόσο η κατανομή αποκλίνει από την κανονική, καθώς έχει δύο κορυφές και δεν είναι συμμετρική, που σημαίνει ότι τα τελευταία δεδομένα δεν προσεγγίζουν την κανονική κατανομή.



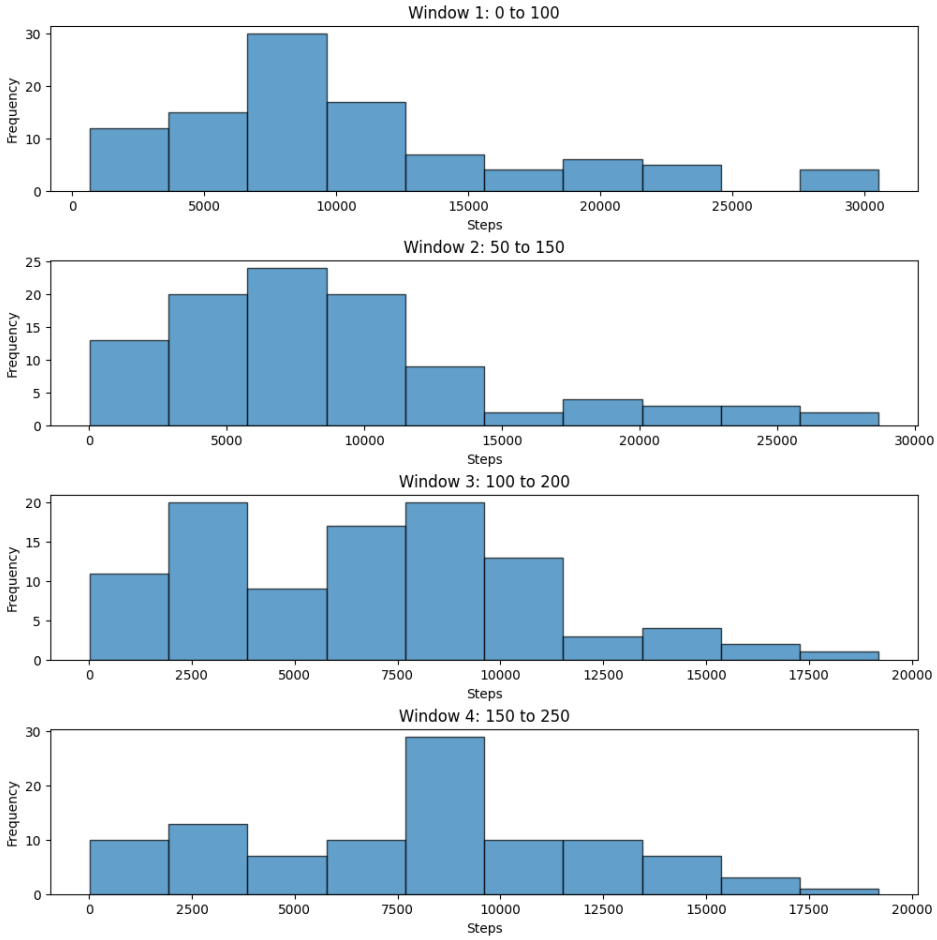
Σχήμα 4-11: Ιστόγραμμα Παραθύρου για συνολικό ύπνο

Όσον αφορά στο ιστόγραμμα εξασθένισης η κατανομή είναι συμμετρική και έχει κορυφή στο διάστημα (450,500), ωστόσο έχει αρκετα μικρότερο εύρος από την κατανομή των αρχικών δεδομένων. Λόγω της μικρής διασποράς δεν προσεγγίζει ικανοποιητικά την κανονική κατανομή



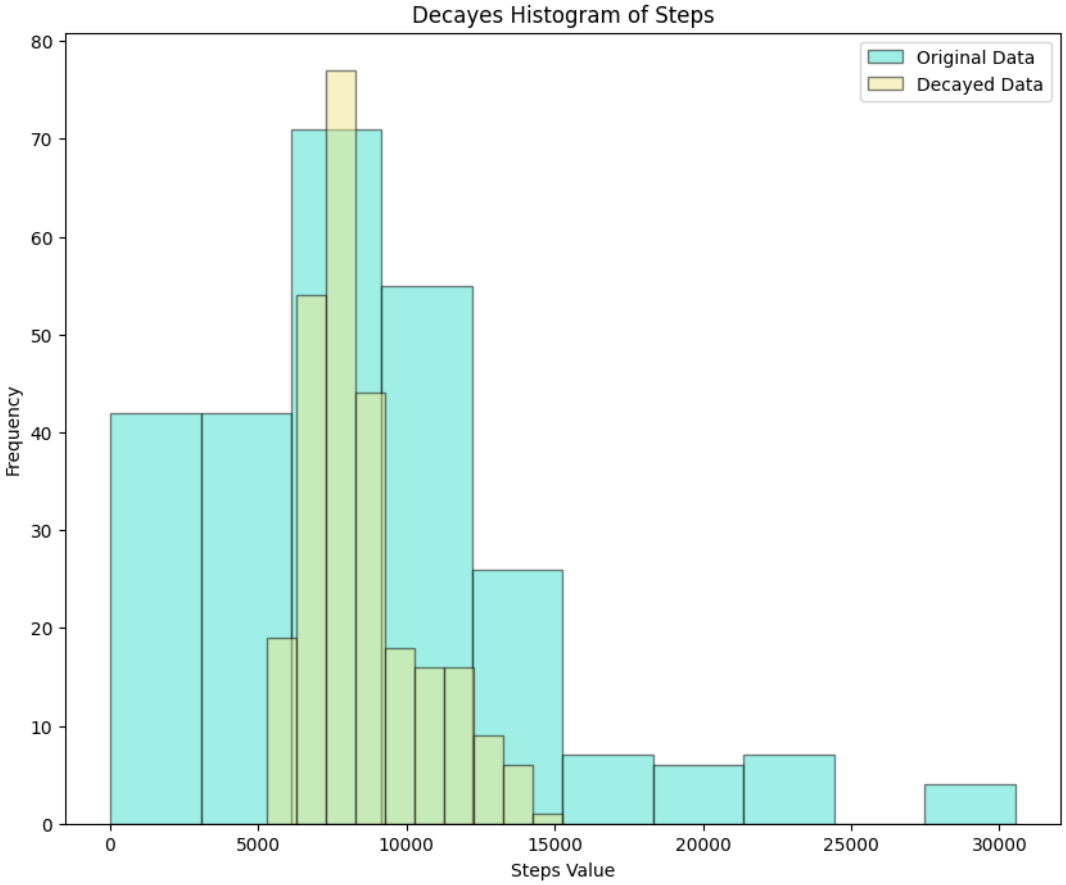
Σχήμα 4-12: Ιστόγραμμα Εξασθένισης για συνολικό ύπνο

Παρακάτω δίνονται τα ιστογράμματα για τα συνολικά βήματα που διήνυσαν οι χρήστες στο διάστημα μίας ημέρας. Στα δύο πρώτα ιστογράμματα παραθύρου έχει μία κορυφή στο διάστημα (5000,10000), ωστόσο δεν είναι συμμετρική. Η πλειοψηφία του πλήθους βημάτων εμφανίζεται στις μικρότερες τιμές, δηλαδή μέχρι 10000, ενώ έχει δεξιά ουρά. Συνεπώς αποκλίνει σημαντικά από την κανονική κατανομή. Στο τρίτο ιστόγραμμα παραθύρου υπάρχουν δύο κορυφές και στο τέταρτο, αν και η κατανομή φαίνεται πιο συμμετρική,έχει επίσης δεξιά ουρά. Σε κανένα από τα ιστογράμματα παραθύρου δεν προσεγγίζεται η κανονική κατανομή.



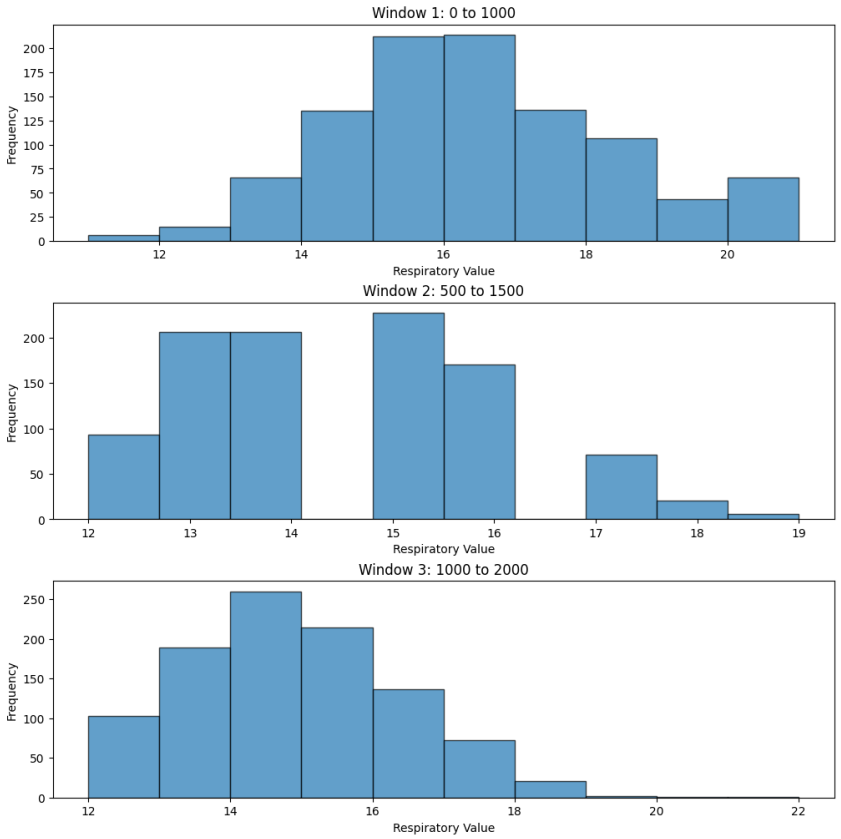
Σχήμα 4-13: Ιστόγραμμα Παραθύρου για βήματα

Στο ιστόγραμμα εξασθένισης η κατανομή είναι συγκεντρωμένη γύρω από τα 10000 βήματα, ωστόσο έχει δεξιά ουρά. Όπως και στα ιστογράμματα παραθύρου οι περισσότερες μετρήσεις καταγράφονται στις χαμηλότερες τιμές. Και εδώ δεν προσεγγίζεται ικανοποιητικά η κανονική κατανομή.



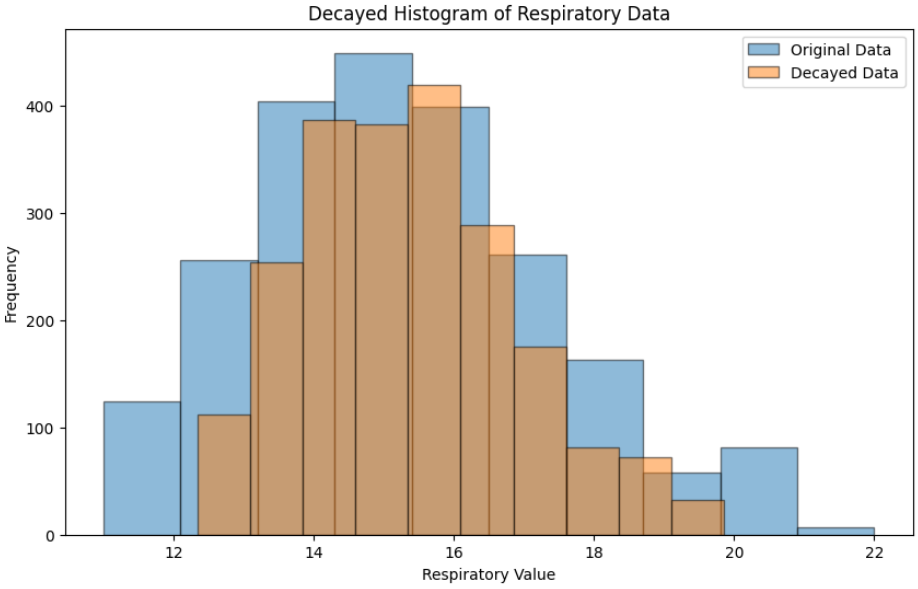
Σχήμα 4-14: Ιστόγραμμα Εξασθένισης για βήματα

Στη συνέχεια δίνονται τα ιστογράμματα για τον αναπνευστικό ρυθμό. Στα ιστογράμματα παραθύρου, με μέγεθος παραθύρου 100, αλλάζει η κατανομή που ακολουθούν οι τιμές. Συγκεκριμένα στο δεύτερο παράθυρο δεν υπάρχουν παρατηρήσεις στα διαστήματα (14,15) και (16,17). Στα ιστογράμματα παραθύρου, η πρώτη κατανομή (Window 1) προσεγγίζει την κανονική κατανομή και είναι συμμετρική γύρω από τη μέση τιμή. Οι άλλες δύο κατανομές παρουσιάζουν ασυμμετρίες, με την τρίτη κατανομή να έχει εμφανή δεξιά ουρά και τη δεύτερη να έχει δύο κορυφές, κάτι που αποκλίνει αρκετά από την κανονική κατανομή.



Σχήμα 4-15: Ιστόγραμμα Παραθύρου για αναπνευστικό ρυθμό

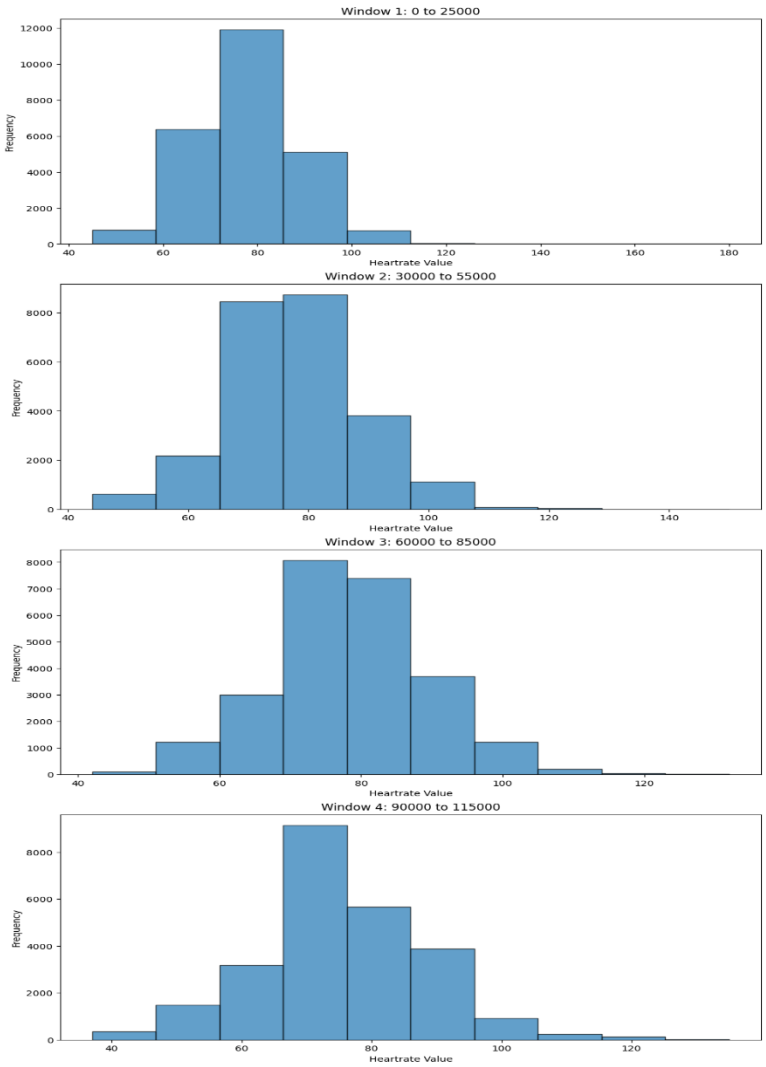
Στο ιστόγραμμα εξασθένισης η κατανομή είναι επίσης συμμετρική, η κορυφή είναι κοντά στην τιμή 16, αλλά η συνολική μορφή της κατανομής έχει ελαφρώς μικρότερη διασπορά σε σύγκριση με τα αρχικά δεδομένα. Τέλος υπάρχει μια ελαφριά δεξιά κλίση, ωστόσο η κατανομή προσεγγίζει την κανονική.



Σχήμα 4-16: Ιστόγραμμα Εξασθένισης για αναπνευστικό ρυθμό

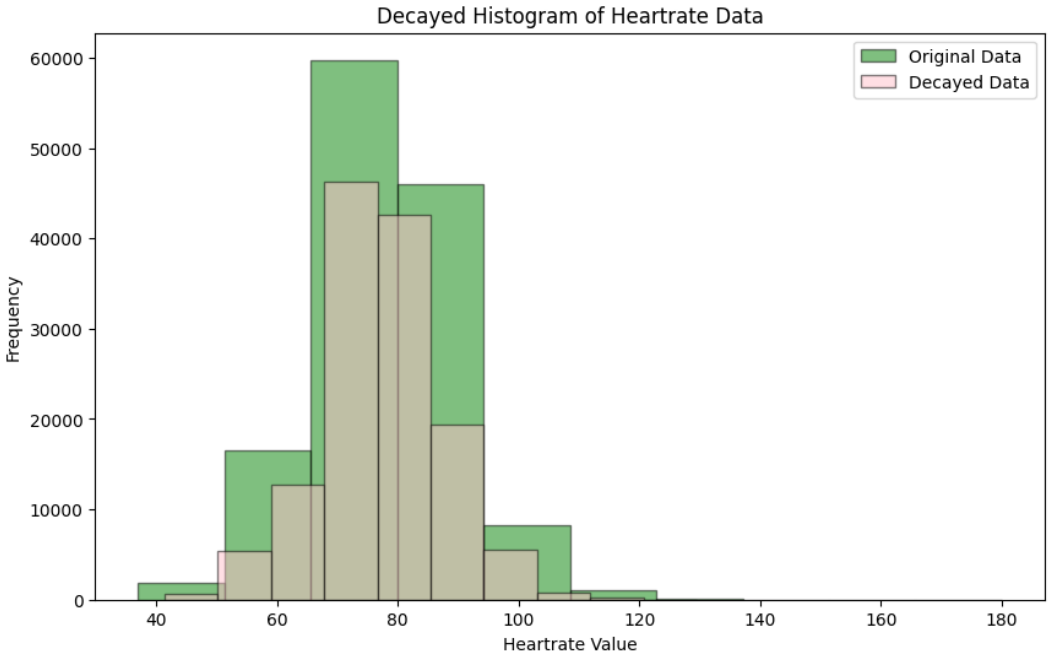
Στα παρακάτω ιστογράμματα αναπαρίστανται τα δεδομένα καρδιακού ρυθμού. Εδώ τα ιστογράμματα παραθύρου και εξασθένισης δίνουν παρόμοια απεικόνιση, καθώς δεν φαίνεται να αλλάζει σημαντικά η συχνότητα εμφάνισης των τιμών στη ροή των δεδομένων.

Στα τέσσερα ιστογράμματα παραθύρου, οι κατανομές είναι συμμετρικές με κορυφή κοντά στην τιμή 80. Υπάρχει μια ελαφριά δεξιά ουρά σε όλες τις κατανομές, αλλά προσεγγίζουν την κανονική κατανομή



Σχήμα 4-17: Ιστόγραμμα Παραθύρου για καρδιακό ρυθμό

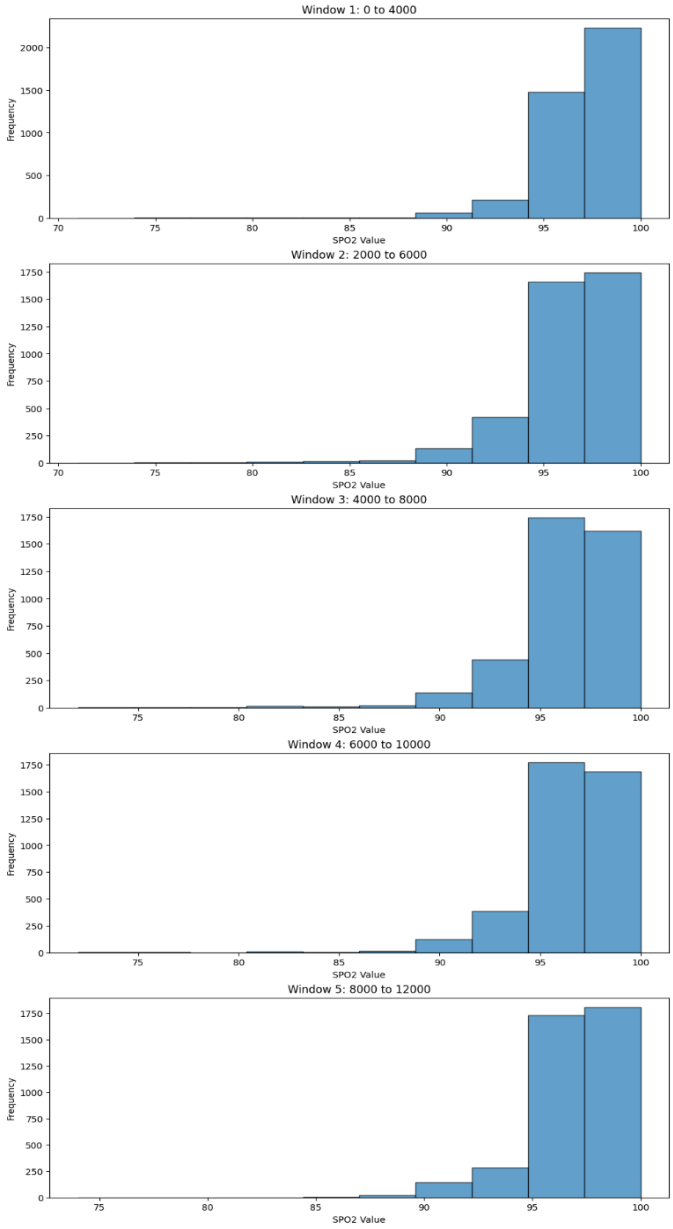
Στο ιστόγραμμα εξασθένισης η κατανομή έχει κορυφή γύρω από τις τιμές 70-80. Φαίνεται να προσεγγίζει την κανονική και είναι πολύ κοντά στα αρχικά δεδομένα . Ωστόσο έχει ελαφρώς δεξιά ουρά.



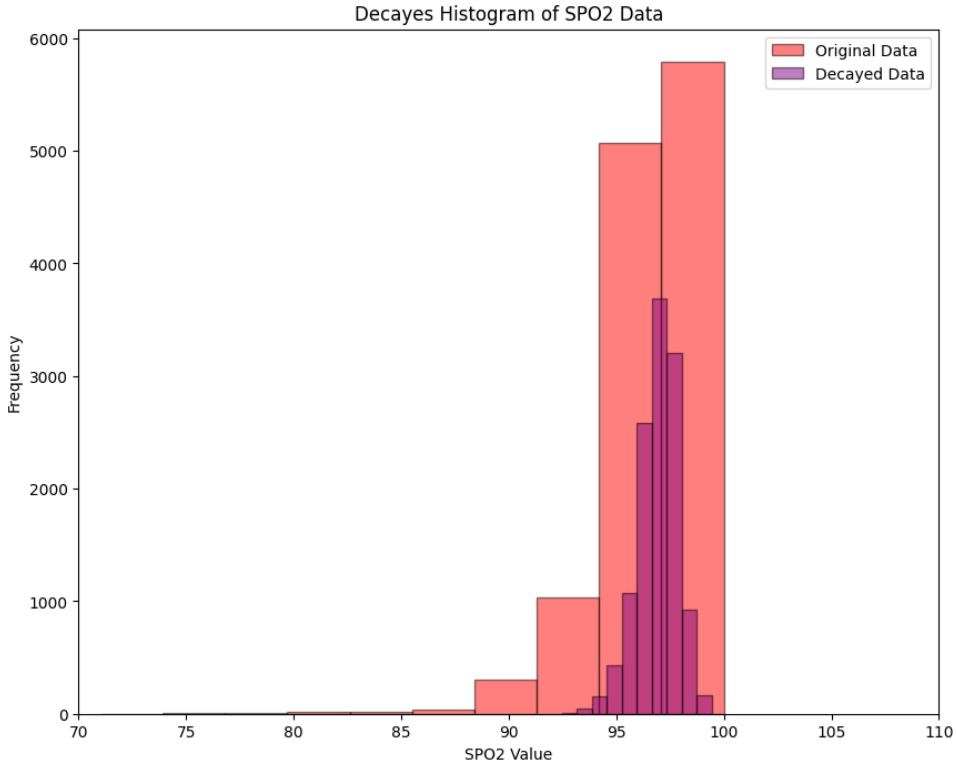
Σχήμα 4-18: Ιστόγραμμα Εξασθένισης για καρδιακό ρυθμό

Παρακάτω δίνονται τα ιστογράμματα για τον περιφερειακό τριχοειδή κορεσμό οξυγόνου. Και εδώ τα ιστογράμματα παραθύρου και εξασθένισης δίνουν παρόμοια απεικόνιση, καθώς δεν φαίνεται να αλλάζει σημαντικά η συχνότητα εμφάνισης των τιμών στη ροή των δεδομένων. Συνεπώς τα επίπεδα οξυγόνου στο αίμα δεν διαφέρουν σημαντικά σε κανένα τμήμα της ροής.

Τόσο στα ιστογράμματα παραθύρου, όσο και στο ιστόγραμμα εξασθένισης, οι τιμές συγκεντρώνονται στο 95% - 100%, κάτι φυσιολογικό καθώς οι περισσότεροι άνθρωποι έχουν επίπεδα οξυγόνου κοντά στο 100%. Οι κατανομές είναι θετικά ασύμμετρες με ελάχιστες τιμές κάτω από 90%. Έχουν μια κορυφή στο διάστημα (95,100) και πολύ μικρές ουρές. Συνεπώς δεν προσεγγίζεται η κανονική κατανομή, καθώς υπάρχει ασυμμετρία, περιορισμένη διασπορά και υψηλή συγκέντρωση των τιμών κοντά στο 100%, κάτι που δεν δηλώνει τυχαία κατανομή γύρω από την κορυφή.

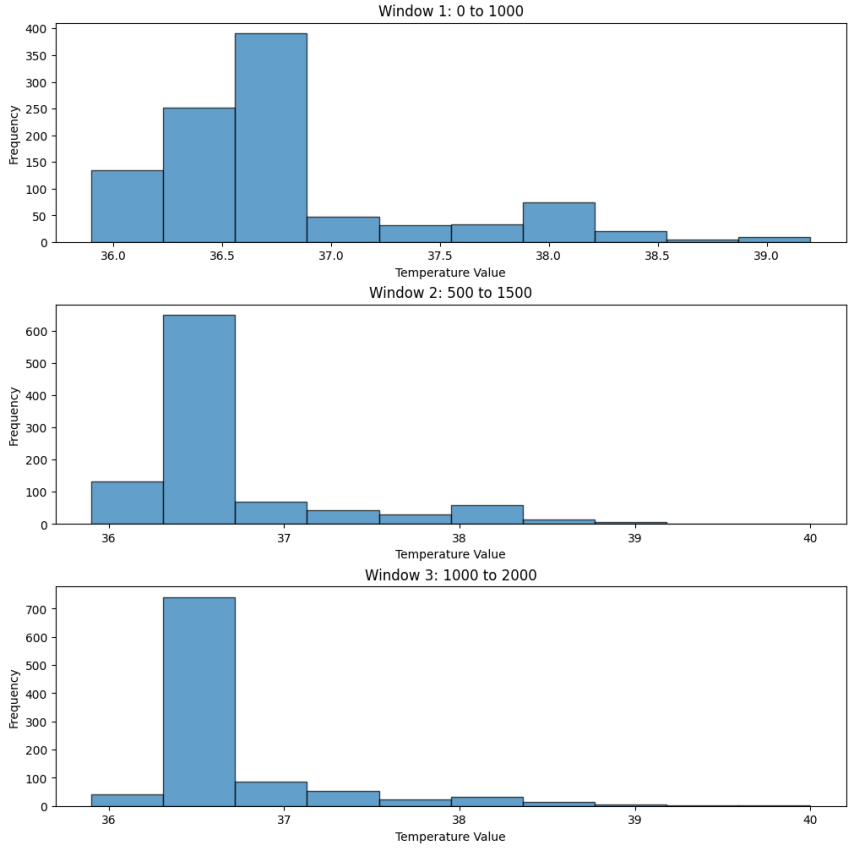


Σχήμα 4-19: Ιστόγραμμα Παραθύρου για SPO2

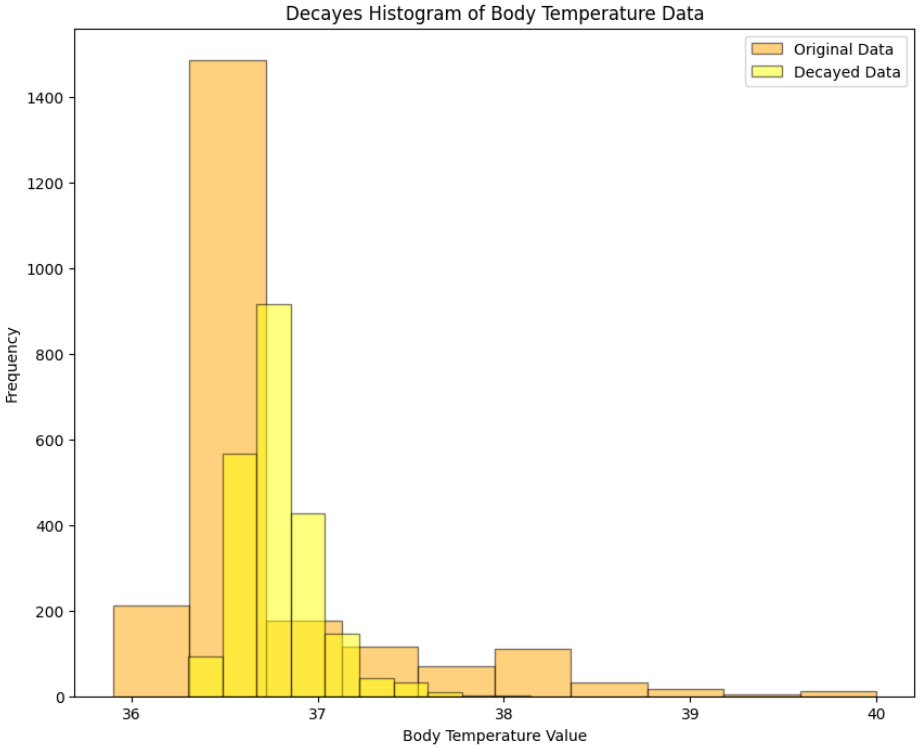


Σχήμα 4-20: Ιστόγραμμα Εξασθένισης για SPO2

Στη συνέχεια δίνονται τα ιστογράμματα για τη θερμοκρασία σώματος. Οι παρατηρήσεις εδώ φαίνεται να παρουσιάζουν μικρότερη διακύμανση στα δύο τελευταία παράθυρα του Ιστογράμματος παραθύρου με επικρατέστερες τις φυσιολογικές θερμοκρασίες κοντά στο 36.6. Ωστόσο στο πρώτο παράθυρο υπάρχουν περισσότερες παρατηρήσεις με υψηλότερες θερμοκρασίες. Η κατανομή είναι ασύμετρη με μεγάλη δεξιά ουρά, προς τις υψηλότερες θερμοκρασίες. Εμφανίζεται επίσης υψηλή συγκέντρωση γύρω από τις φυσιολογικές θερμοκρασίες (36°C - 37°C). Συνεπώς η κατανομή δεν είναι κανονική.



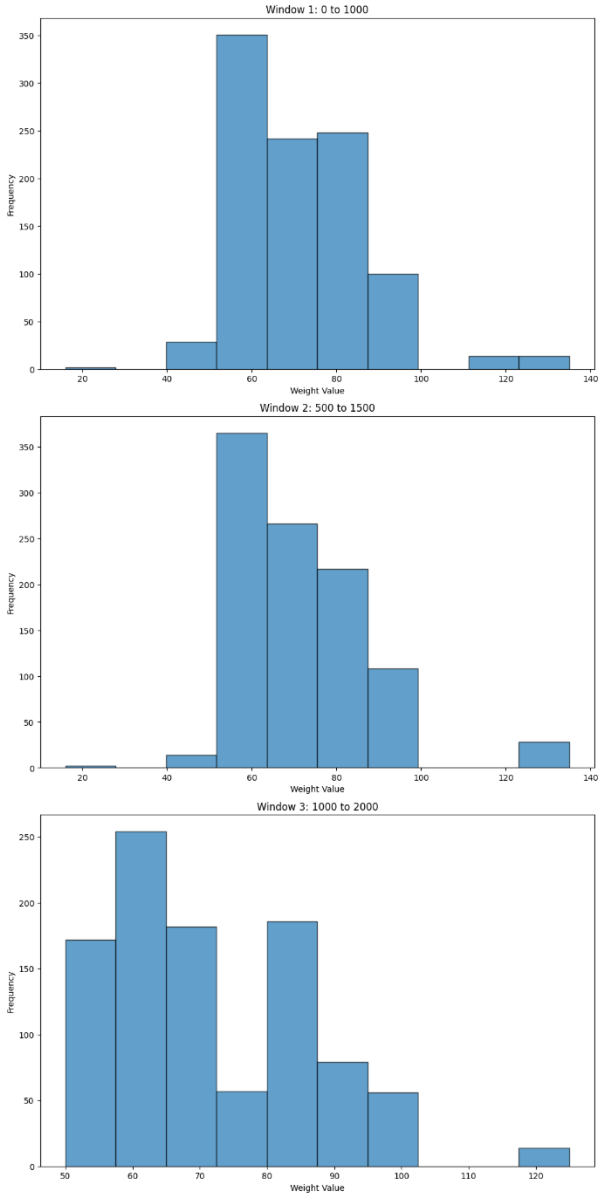
Σχήμα 4-21: Ιστόγραμμα Παραθύρου για θερμοκρασία σώματος



Σχήμα 4-22: Ιστόγραμμα Εξασθένισης για θερμοκρασία σώματος

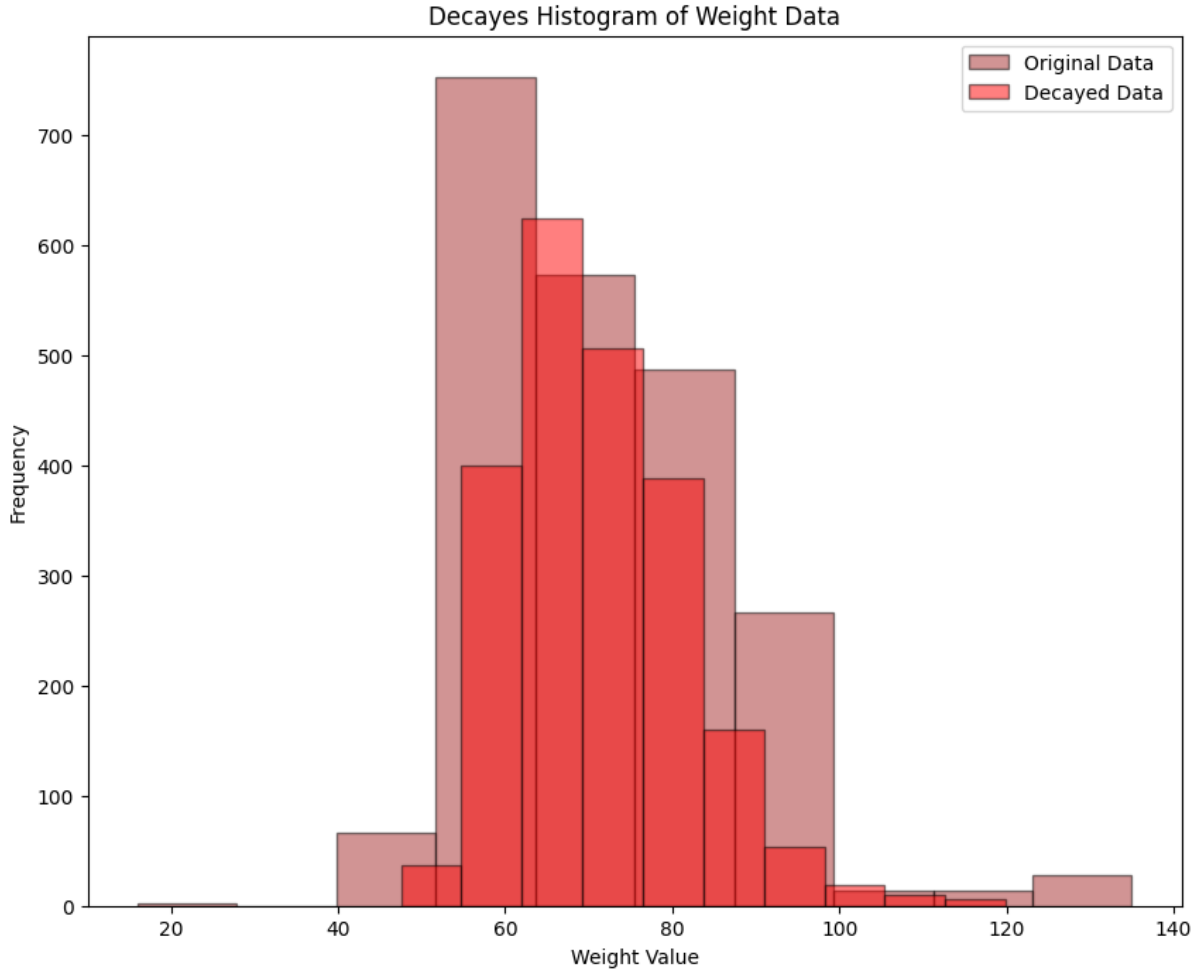
Τέλος δίνονται τα ιστογράμματα για το σωματικό βάρος. Όπως και στην περίπτωση της θερμοκρασίας του σώματος, παρατηρούνται μεταβολές στις συχνότητες των τιμών στα τρία παράθυρα. Αυτό υποδηλώνει αλλαγή στη συχνότητα των τιμών του σωματικού βάρους στα διάφορα παράθυρα. Υπάρχει δηλαδή μεγάλο εύρος τιμών που ποικίλει στα τρία παράθυρα.

Στα ιστογράμματα παραθύρου, η κατανομή έχει κορυφή στο διάστημα (60,70) και είναι ασύμμετρη, καθώς έχει μεγάλη δεξιά ουρά, γεγονός που προκαλεί απόκλιση από την κανονική κατανομή.



Σχήμα 4-23: Ιστόγραμμα Παραθύρου για σωματικό βάρος

Στο ιστόγραμμα συχνότητας, η κατανομή παραμένει ασύμμετρη, καθώς οι τιμές δεν είναι εξίσου κατανεμημένες γύρω από τη μέση τιμή. Υπάρχει επίσης ουρά προς τα δεξιά και οι τιμές είναι συγκεντρωμένες στο διάστημα (60,70). Συνεπώς δεν προσεγγίζεται η κανονική κατανομή.



Σχήμα 4-24: Ιστόγραμμα Εξασθένισης για σωματικό βάρος

Από τα ιστογράμματα προκύπτουν τα ακόλουθα συμπεράσματα:

- Τα δεδομένα συνολικού ύπνου αναμένεται να προσεγγίζουν την κανονική κατανομή, καθώς ο ύπνος εξαρτάται από πολλούς παράγοντες, κάτι που επαληθεύεται και στα περισσότερα διαγράμματα. Δεν βγαίνει λοιπόν κάποιο συμπέρασμα.
- Στα συνολικά βήματα δεν προσεγγίζεται η κανονική κατανομή. Παρατηρείται μεγάλη συγκέντρωση σε μικρό αριθμό βημάτων, κάτι που εξηγείται είτε από την

εξασθένιση της φυσικής κατάστασης των ασθενών με Covid-19 είτε λόγω του περιορισμού ως μέτρο πρόληψης για την αποφυγή μετάδωσης του ιού.

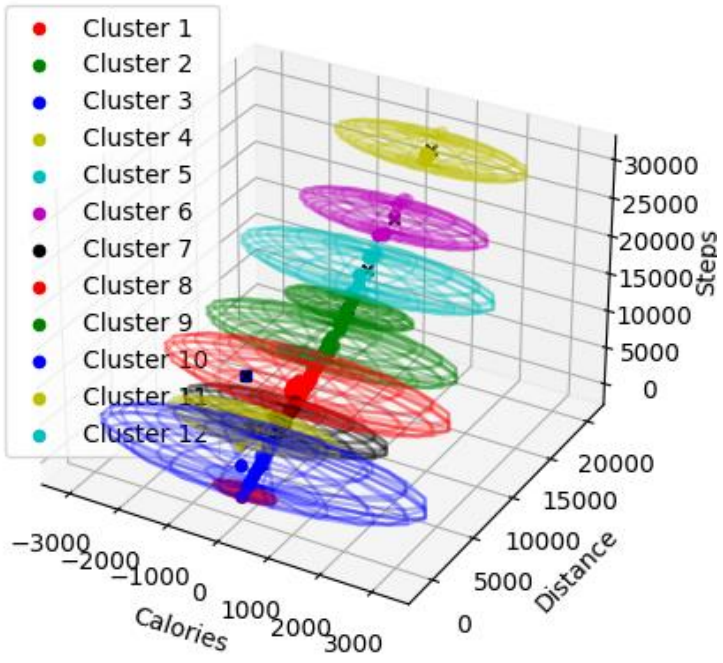
- Ο αναπνευστικός ρυθμός υπό φυσιολογικές συνθήκες ακολουθεί την κανονική κατανομή, καθώς εξαρτάται από πολλούς παράγοντες. Στα δεδομένα της παρούσας διπλωματικής ωστόσο, η κατανομή έχει ελαφριά δεξιά ουρά. Από το γεγονός αυτό ερμηνεύεται ότι ενώ οι περισσότεροι ασθενείς έχουν ήπια ανεπνευστικά συμπτώματα, ορισμένοι αντιμετωπίζουν δυσκολία στην αναπνοή, ή έχουν ταχύπνοια.
- Στον καρδιακό ρυθμό παρατηρείται επίσης ελαφριά δεξιά ουρά. Συνεπώς υπάρχει ένας μικρός αριθμός ασθενών, των οποίων το καρδιαγγειακό σύστημα έχει επηρεαστεί από τον ιό, προκαλώντας ταχυκαρδία.
- Ο περιφερειακός τριχοειδής κορεσμός οξυγόνου δεν ακολουθεί κανονική κατανομή, καθώς τα φυσιολογικά όρια είναι στο (95-100). Ωστόσο παρατηρούνται μερικές χαμηλές τιμές, ακόμα και κάτω από 90 που πιθανόν να οφείλονται στον Covid-19.
- Η θερμοκρασία υπό φυσιολογικές συνθήκες δεν ακολουθεί κανονική κατανομή, καθώς οι φυσιολογικές τιμές βρίσκονται στο διάστημα (36,37). Στα ιστογράμματα εντοπίζονται ορισμένες παρατηρήσεις με τιμές μεγαλύτερες του 37, κάτι που υποδηλώνει πως υπάρχουν ασθενείς με πυρετό.
- Το σωματικό βάρος υπό φυσιολογικές συνθήκες επίσης δεν ακολουθεί την κανονική κατανομή, καθώς η πλειοψηφία είναι κοντά στη μέση τιμή. Στα ιστογράμματα παρατηρούνται ορισμένες πολύ υψηλές τιμές, οι οποίες ίσως να υποδηλώνουν υπέρβαρα άτομα.

4.5 Συσταδοποίηση - Αλγόριθμος DBIECM

Στο κεφάλαιο αυτό θα εφαρμοστεί ο αλγόριθμος DBIECM στο σύνολο των δεδομένων. Παράλληλα θα υπολογιστεί το Silhouette Score και θα εκτιμηθεί το καλύτερο *Dthr* για την

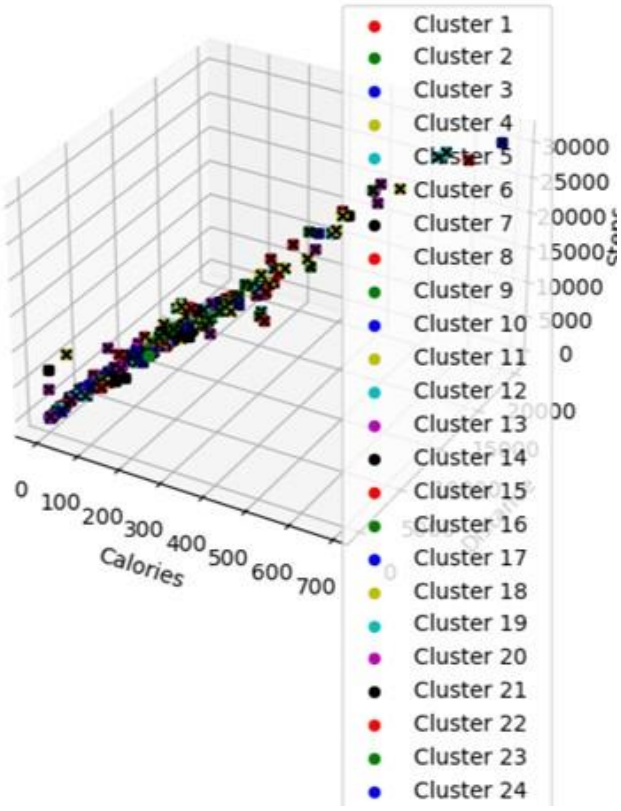
αποδοτικότερη συσταδοποίηση των δεδομένων. Υπενθυμίζεται πως ο αλγόριθμος DBIECM είναι ένας αλγόριθμος συσταδοποίησης που βασίζεται στην απόσταση και απαιτεί από το χρήστη να ορίσει την μέγιστη ακτίνα συστάδας (*Dthr*), κατατάσσοντας τα δεδομένα σε σφαιρικές συστάδες με βάση την ομοιότητά τους. Το Silhouette score είναι ένα μέτρο που χρησιμοποιείται για να αξιολογήσει την ποιότητα των συστάδων που δημιουργούνται από έναν αλγόριθμο συσταδοποίησης, μετρώντας πόσο όμοιο είναι ένα στοιχείο με τα υπόλοιπα στοιχεία της συστάδας στην οποία ανήκει σε σύγκριση με άλλες συστάδες.

Παρακάτω δίνεται το γράφημα που προκύπτει από τη συσταδοποίηση των δεδομένων που αφορούν στην καθημερινή δραστηριότητα, δηλαδή τα βήματα, την απόσταση που έχουν διανύσει και τις θερμίδες που έχουν καταναλώσει οι ασθενείς. Τα δεδομένα αυτά όπως είναι φυσικό έχουν γραμμική εξάρτηση μεταξύ τους, κάτι που φαίνεται και στο γράφημα. Γίνεται επίσης εύκολα αντιληπτό ότι η συσταδοποίηση που έχει γίνει δεν είναι η αποδοτικότερη και αρκετά δεδομένα είναι κοντά στο όριο απόφασης μεταξύ δύο συστάδων.

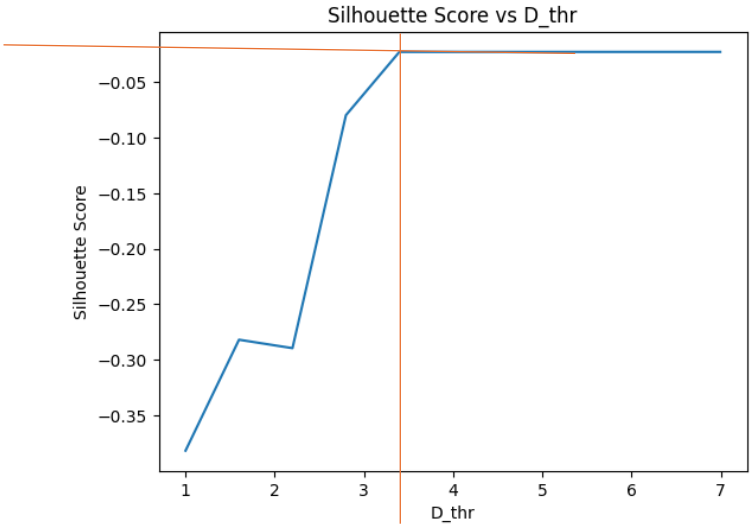


Σχήμα 4-25: Αποτελέσματα συσταδοποίησης για τα δεδομένα καθημερινής δραστηριότητας με *Dthr* 3000

Υπολογίζοντας το Silhouette Score για το διάστημα τιμών *Dthr* 1 έως 3000 προκύπτει ότι καλύτερα αποτελέσματα συσταδοποίησης θα δώσει η τιμή 3.4. Ωστόσο βάζοντας ως όριο την τιμή αυτή, το σύνολο των δεδομένων διαχωρίζεται σε πάρα πολλές συστάδες, συγκεκριμένα σε 212, ενώ για το συγκεκριμένο όριο το silhouette score είναι πολύ κοντά στο 0 (συγκεκριμένα -0.23), όπως φαίνεται στο διάγραμμα με τις τιμές Silhouette Scores παρακάτω. Όπως έχει αναφερθεί και στο κεφάλαιο 3, τιμή κοντά στο 0 υποδηλώνει ότι οι συστάδες δεν είναι καλά διαχωρισμένες και τα δείγματα είναι κοντά στο όριο απόφασης μεταξύ συστάδων.Συνεπώς οι συστάδες εξακολουθούν να μην είναι καλά διαχωρισμένες και αρκετά σημεία είναι κοντά στο όριο απόφασης μεταξύ συστάδων.



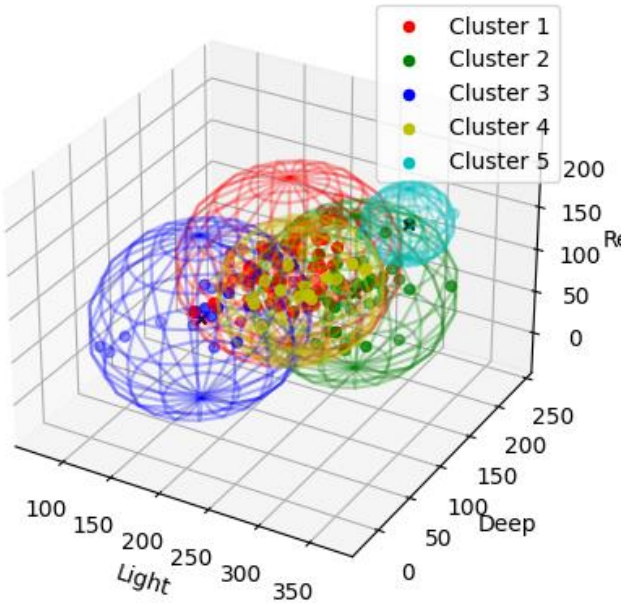
Σχήμα 4-26: Αποτελέσματα συσταδοποίησης για τα δεδομένα καθημερινής δραστηριότητας με *Dthr* 3.4



Σχήμα 4-27: Τιμές Silhouette score για τα δεδομένα καθημερινής δραστηριότητας

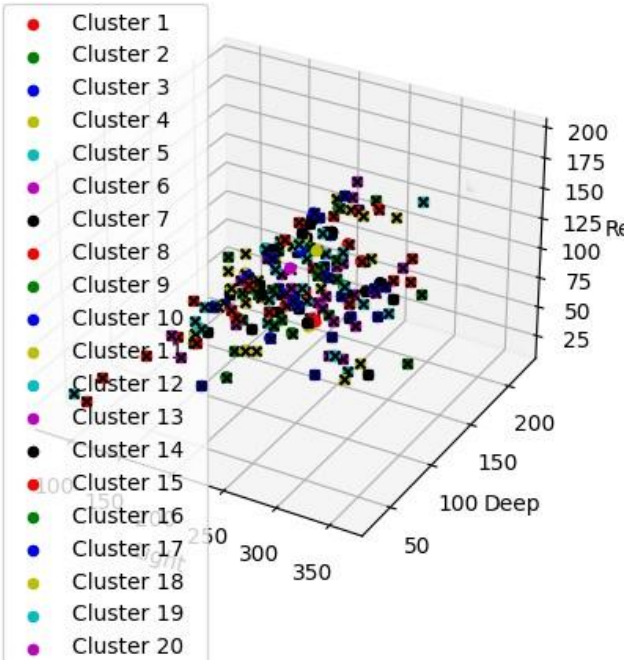
Στο παραπάνω γράφημα φαίνεται η τιμή -0.23 για $Dthr = 3.4$. Γίνεται επίσης αντιληπτό πώς η τιμή του Silhouette score σταματάει να αυξάνεται μετά την τιμή 3.4.

Στη συνέχεια εφαρμόζεται ο αλγόριθμος DBIECM στα δεδομένα ύπνου και συγκεκριμένα στα λεπτά ελαφρύ, βαθύ και ύπνου rem. Τα αποτελέσματα που προκύπτουν είναι τα ακόλουθα:

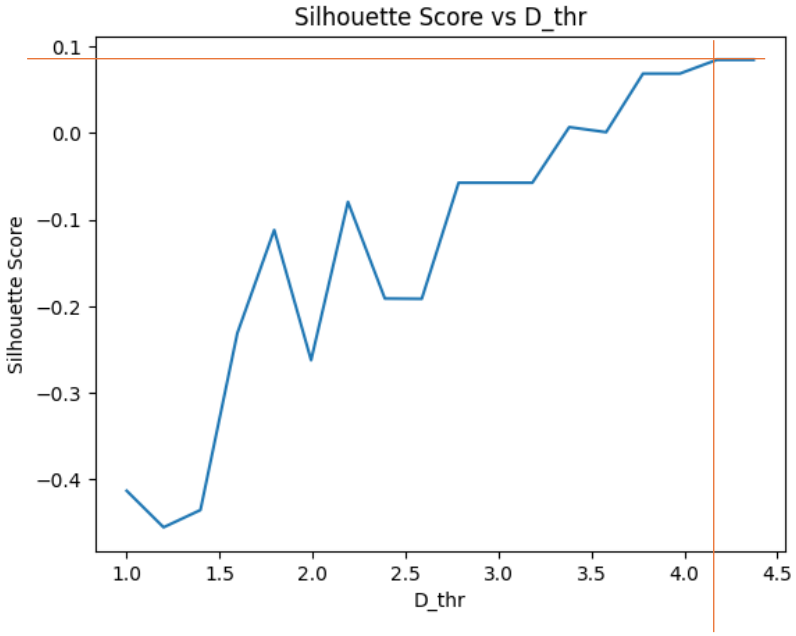


Σχήμα 4-28: Αποτελέσματα συσταδοποίησης για τα δεδομένα ύπνου με *Dthr* 100

Υπολογίζοντας το Silhouette Score για το διάστημα τιμών Dthr 1 έως 100 προκύπτει ότι καλύτερα αποτελέσματα συσταδοποίησης θα δώσει η τιμή 4.2 για την ακτίνα. Ωστόσο βάζοντας ως όριο την τιμή αυτή, το σύνολο των δεδομένων διαχωρίζεται σε πάρα πολλές συστάδες, συγκεκριμένα σε 178, ενώ για το συγκεκριμένο όριο το silhouette score είναι λίγο παραπάνω από την τιμή 0 (συγκεκριμένα 0.085). Συνεπώς οι συστάδες συνεχίζουν να μην είναι καλά διαχωρισμένες και αρκετά σημεία είναι κοντά στο όριο απόφασης μεταξύ συστάδων. Τα δεδομένα ύπνου ωστόσο διαχωρίζονται λίγο καλύτερα από τα δεδομένα καθημερινής δραστηριότητας, όπως φαίνεται και στο παρακάτω γράφημα.



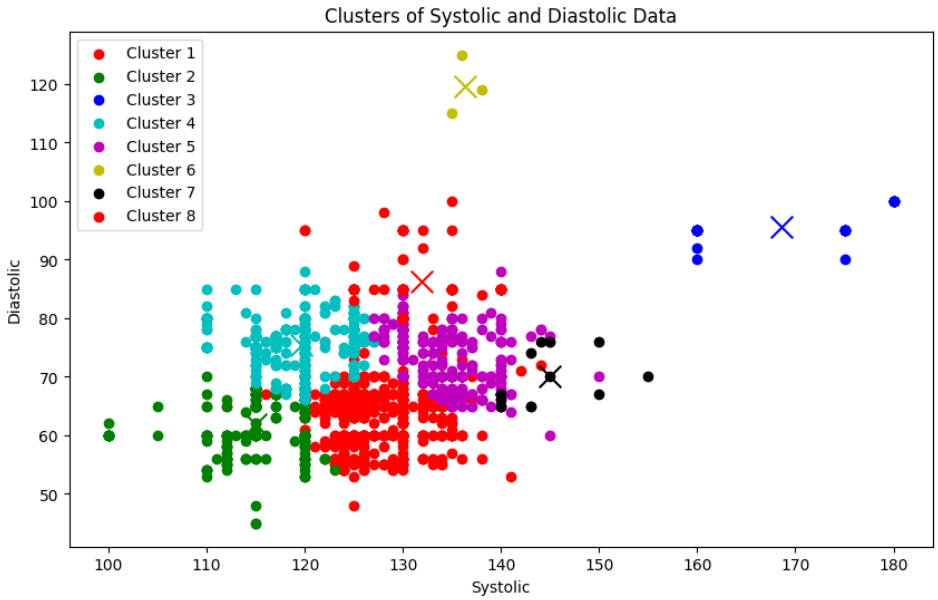
Σχήμα 4-29: Αποτελέσματα συσταδοποίησης για τα δεδομένα ύπνου με *Dthr* 4.2



Σχήμα 4-30: Τιμές Silhouette score για τα δεδομένα ύπνου

Στο παραπάνω γράφημα φαίνεται η τιμή 0.085 για $Dthr = 4.2$. Γίνεται επίσης αντιληπτό πώς η τιμή του Silhouette score σταματάει να αυξάνεται μετά την τιμή 4.2.

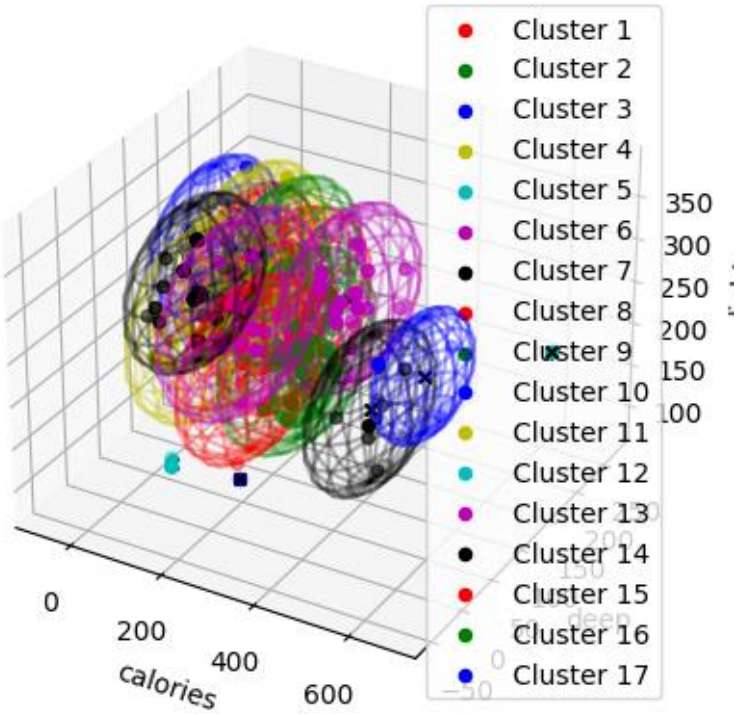
Στη συνέχεια εφαρμόζεται ο αλγόριθμος DBIECM στις μετρήσεις βιοσημάτων και συγκεκριμένα στη διαστολική και στη συστολική πίεση και προκύπτουν τα ακόλουθα αποτελέσματα:



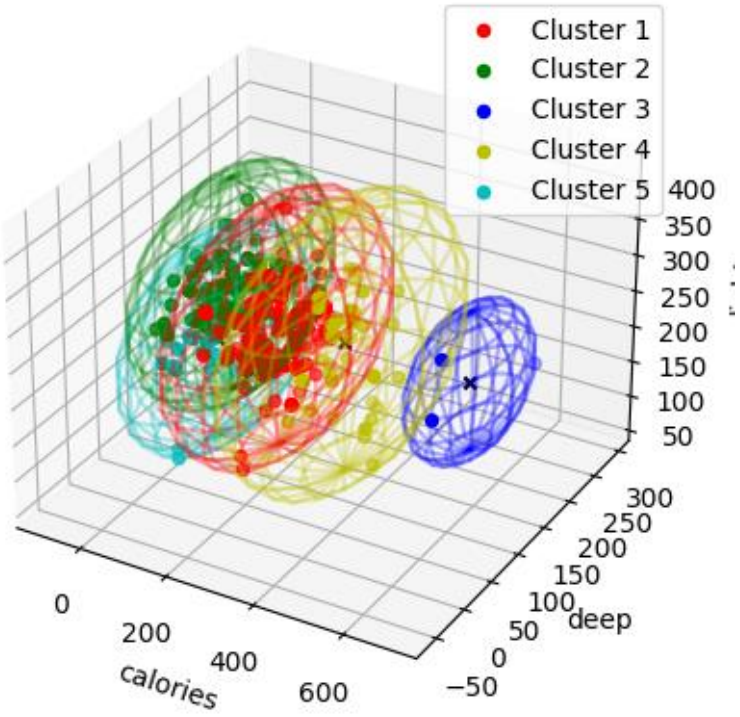
Σχήμα 4-31: Αποτελέσματα συσταδοποίησης για τα δεδομένα ύπνου με *Dthr* 20

Οι συστάδες σε μεγάλο βαθμό είναι διακριτές. Κάθε συστάδα συγκεντρώνεται σε διαφορετικά σημεία του διαγράμματος και υπάρχει μικρή επικάλυψη μεταξύ των περισσότερων. Πιο συγκεκριμένα, η Cluster 1 (κόκκινη), που περιέχει πολλά δεδομένα με μικρή διασπορά, διαχωρίζεται από τις Cluster 2 (γαλάζια) και Cluster 3 (μπλε) . Ωστόσο υπάρχει μερική επικάλυψη των Cluster 1 (κόκκινη) και Cluster 4 (μωβ) καθώς και μεταξύ των Cluster 2 (γαλάζια) και Cluster 5 (πράσινη). Η Cluster 6 (κίτρινη), βρίσκεται μακριά από τις υπόλοιπες συστάδες και έχει μικρό πλήθος. Τα δεδομένα της θα μπορούσαν να θεωρηθούν ακραίες τιμές. Η συσταδοποίηση είναι σε γενικές γραμμές αποτελεσματική, καθώς έχει διαχωρίσει τα δεδομένα σε αρκετές διακριτές συστάδες. Παρόλο που υπάρχει κάποια αλληλεπικάλυψη μεταξύ συστάδων με κοντινές τιμές, το μοντέλο καταφέρνει να αναγνωρίσει συστάδες και να εντοπίσει ακραίες τιμές.

Στη συνέχεια συγχωνεύονται τα σύνολα δεδομένων καθημερινής δραστηριότητας βημάτων και ύπνου. Πιο συγκεκριμένα συγχωνεύονται ο ελαφρύς και βαθύς ύπνος με τις θερμίδες και εφαρμόζεται ο αλγόριθμος DBIECM. Προκύπτουν τα ακόλουθα αποτελέσματα:



Σχήμα 4-32: Αποτελέσματα συσταδοποίησης για τα δεδομένα ύπνου και βημάτων με *Dthr* 100



Σχήμα 4-33: Αποτελέσματα συσταδοποίησης για τα δεδομένα ύπνου και βημάτων με *Dthr* 200

Για *Dthr* 100 παρατηρείται σημαντική επικάλυψη μεταξύ των συστάδων, ειδικότερα στο κέντρο του διαγράμματος, που σημαίνει ότι δεν γίνεται σωστός διαχωρισμός των δεδομένων. Προκειμένου να βελτιωθεί η απόδοση του αλγορίθμου αυξάνεται το *Dthr* ώστε να μειωθεί ο αριθμός των συστάδων. Για *Dthr* 200 υπάρχει βελτίωση της αποτελεσματικότητας του αλγορίθμου. Η επικάλυψη έχει μειωθεί , ωστόσο εξακολουθεί να υπάρχει σε σημαντικό βαθμό μεταξύ μεταξύ των Cluster 1 (κόκκινη), Cluster 2 (πράσινη), Cluster 4 (κίτρινη) και Cluster 5 (γαλάζια). Η επικάλυψη αυτή πιθανόν να οφείλεται σε ομοιότητα των δεδομένων, ή τα χαρακτηριστικά που χρησιμοποιήθηκαν δεν επαρκούν για τον πλήρη διαχωρισμό των συστάδων

Ως γενικό συμπέρασμα, προκύπτει ότι τα δεδομένα, με εξαίρεση τη συστολική και διαστολική πίεση δεν είναι εύκολα διαχωρίσιμα και είναι αρκετά όμοια. Ενδεχομένως ένας διαφορετικός αλγόριθμος συσταδοποίησης να ήταν αποτελεσματικότερος.

4.6 Εύρεση συχνών μοτίβων

Στο κεφάλαιο αυτό θα εφαρμοστεί ο αλγόριθμος των Gurmeet Singh Manku και Rajeev Motwani. Όπως αναλύθηκε εκτενώς σε προηγούμενο κεφάλαιο ο αλγόριθμος αυτός είναι μια προσέγγιση εξόρυξης συχνών μοτίβων σε ροές δεδομένων. Στον αλγόριθμο που εκτελείται για τους σκοπούς της παρούσας διπλωματικής χρησιμοποιείται η δομή δεδομένων TRIE για την αποθήκευση και επεξεργασία των μοτίβων, τα οποία εμφανίζονται στο output μαζί με το πλήθος εμφάνισής τους, στα δεδομένα ύπνου και στα δεδομένα πίεσης. Τα αποτελέσματα που προκύπτουν βρίσκονται παρακάτω.

α)Συχνά Μοτίβα στα δεδομένα ύπνου

Το output της python μας δίνει τα παρακάτω:

```
Μοτίβο:{(0, 294): <__main__.TrieNode object at 0x79b9574c4550>,(1, 44): <__main__.TrieNode
object at 0x79b9574d3a90>,(2, 154):<__main__.TrieNode object at 0x79b9574d3af0>}, Συχνότητα: 5
Μοτίβο:{(0, 208): <__main__.TrieNode object at 0x79b9574c5c30>,(1, 133): <__main__.TrieNode
object at 0x79b9574d3850>,(2, 83):<__main__.TrieNode object at 0x79b9574d38b0>}, Συχνότητα: 4
Μοτίβο:{(0, 189): <__main__.TrieNode object at 0x79b9574c7490>,(1, 114): <__main__.TrieNode
object at 0x79b9574d3550>,(2, 107):<__main__.TrieNode object at 0x79b9574d35b0>}, Συχνότητα: 5
Μοτίβο:{(0, 215): <__main__.TrieNode object at 0x79b9574d0370>,(1, 194): <__main__.TrieNode
object at 0x79b9574d39d0>,(2, 120):<__main__.TrieNode object at 0x79b9574d3a30>}, Συχνότητα: 5
Μοτίβο:{(0, 206): <__main__.TrieNode object at 0x79b9574d2650>,(1, 97): <__main__.TrieNode
object at 0x79b9574d3490>,(2, 119):<__main__.TrieNode object at 0x79b9574d34f0>}, Συχνότητα:4
Μοτίβο:{(0, 238): <__main__.TrieNode object at 0x79b9574d2890>,(1, 177): <__main__.TrieNode
object at 0x79b9574d3910>,(2, 105):<__main__.TrieNode object at 0x79b9574d3970>}, Συχνότητα: 4
Μοτίβο:{(0, 175): <__main__.TrieNode object at 0x79b9574d3610>,(1, 154): <__main__.TrieNode
object at 0x79b9574d3670>,(2, 100):<__main__.TrieNode object at 0x79b9574d36d0>}, Συχνότητα: 4
Μοτίβο:{(0, 285): <__main__.TrieNode object at 0x79b9574d3730>,(1, 151): <__main__.TrieNode
object at 0x79b9574d3790>,(2, 92):<__main__.TrieNode object at 0x79b9574d37f0>}, Συχνότητα: 5
```

Ο κώδικας εντοπίζει μοτίβα που εμφανίζονται συχνά σύμφωνα με το κατώφλι υποστήριξης που ορίζεται στο 2%. Στη δομή δεδομένων TRIE του αλγορίθμου καταγράφεται η συχνότητα εμφάνισης κάθε συνδυασμού και επιστρέφει αυτά που υπερβαίνουν το κατώφλι. Τα αποτελέσματα αυτά συνοψίζονται στον παρακάτω πίνακα:

Ελαφρύς ύπνος	Βαθύς ύπνος	REM ύπνος	Αριθμός εμφάνισης
294	44	154	5
208	133	83	4

189	114	107	5
215	194	120	5
206	97	119	4
238	177	105	4
175	154	100	4
285	151	92	5

Πίνακας 4-2: Συχνά μοτίβα στα δεδομένα ύπνου

Από τον παραπάνω πίνακα προκύπτει ότι ο ελαφρύς ύπνος εμφανίζεται με διακυμάνσεις από 175 έως 294, κάτι που είναι λογικό, καθώς υπό φυσιολογικές συνθήκες αποτελεί το μεγαλύτερο ποσοστό του ύπνου. Ο βαθύς ύπνος κυμαίνεται από 44 έως 194, με αρκετά μεγάλες διακυμάνσεις που πιθανόν οφείλονται σε διαφορους παράγοντες όπως η ηλικία η φυσική κατάσταση, αλλά και η υγεία. Συνεπώς ίσως να υπάρχει ένδειξη ότι ο Covid-19 επηρεάζει το ποσοστό βαθιού ύπνου. Τέλος ο ύπνος rem κυμαίνεται από 83 έως 154. Στα μοτίβα εμφανίζεται σταθερή εναλλαγή ανάμεσα στις διαφορετικές φάσεις του ύπνου. Οι φάσεις με μεγαλύτερη διάρκεια ελαφρού ύπνου συνδυάζονται με χαμηλότερες τιμές βαθιού ύπνου, ενώ ο ύνος rem εμφανίζει τη μικρότερη διακύμανση. Στα περισσότερα συχνά μοτίβα ο βαθύς ύπνος και ο ύπνος rem εμφανίζονται σε μεγάλο ποσοστό, κάτι που υποδηλώνει καλή ποιότητα ύπνου.

Στο σύνολο του, ο πίνακας δίνει μια εικόνα σταθερών αλλά ποικιλόμορφων μοτίβων ύπνου με τις αναμενόμενες διακυμάνσεις ανάμεσα σε ελαφρύ, βαθύ και ύπνο rem. Τα μοτίβα αυτά θα μπορούσαν να χρησιμοποιηθούν για περαιτέρω ανάλυση της ποιότητας του ύπνου. Για παράδειγμα, θα μπορούσαν εντοπισθούν προβλήματα ύπνου και να προταθούν βέλτιστες ώρες ύπνου για τον κάθε χρήστη, αναλύοντας εξατομικευμένα τα δεδομένα του. Παράλληλα μπορεί να γίνει εξαγωγή κανόνων συσχέτισης με άλλες παραμέτρους υγείας, όπως η αρτηριακή πίεση της οποίας τα συχνά μοτίβα εντοπίζονται παρακάτω.

β)Συχνά Μοτίβα στα δεδομένα πίεσης

Το output της python μας δίνει τα παρακάτω:

```
Συχνά μοτίβα πίεσης:
Μοτίβο: {(0, 125.0): <__main__.TrieNode object at 0x79b9468cfee0>, (1, 75.0):
<__main__.TrieNode object at 0x79b9472ea4a0>}, Συχνότητα: 33
Μοτίβο: {(0, 125.0): <__main__.TrieNode object at 0x79b9468cfee0>, (1, 80.0):
<__main__.TrieNode object at 0x79b94e4bf460>}, Συχνότητα: 40
Μοτίβο: {(0, 130.0): <__main__.TrieNode object at 0x79b948ba3bb0>, (1, 70.0):
<__main__.TrieNode object at 0x79b946e2df60>}, Συχνότητα: 55
Μοτίβο: {(0, 130.0): <__main__.TrieNode object at 0x79b948ba3bb0>, (1, 65.0):
<__main__.TrieNode object at 0x79b9574e2f50>}, Συχνότητα: 44
Μοτίβο: {(0, 130.0): <__main__.TrieNode object at 0x79b948ba3bb0>, (1, 67.0):
<__main__.TrieNode object at 0x79b947852170>}, Συχνότητα: 39
Μοτίβο: {(0, 130.0): <__main__.TrieNode object at 0x79b948ba3bb0>, (1, 60.0):
<__main__.TrieNode object at 0x79b9472ebbe0>}, Συχνότητα: 56
Μοτίβο: {(0, 130.0): <__main__.TrieNode object at 0x79b948ba3bb0>, (1, 80.0):
<__main__.TrieNode object at 0x79b947374520>}, Συχνότητα: 64
Μοτίβο: {(0, 115.0): <__main__.TrieNode object at 0x79b947a337f0>, (1, 65.0):
<__main__.TrieNode object at 0x79b947a33850>}, Συχνότητα: 48
Μοτίβο: {(0, 115.0): <__main__.TrieNode object at 0x79b947a337f0>, (1, 75.0):
<__main__.TrieNode object at 0x79b9464afe20>}, Συχνότητα: 69
Μοτίβο: {(0, 117.0): <__main__.TrieNode object at 0x79b947824070>, (1, 77.0):
<__main__.TrieNode object at 0x79b9464afac0>}, Συχνότητα: 33
Μοτίβο: {(0, 120.0): <__main__.TrieNode object at 0x79b947862bf0>, (1, 60.0):
<__main__.TrieNode object at 0x79b947862c50>}, Συχνότητα: 35
Μοτίβο: {(0, 120.0): <__main__.TrieNode object at 0x79b947862bf0>, (1, 65.0):
<__main__.TrieNode object at 0x79b9478a0430>}, Συχνότητα: 35
Μοτίβο: {(0, 120.0): <__main__.TrieNode object at 0x79b947862bf0>, (1, 70.0):
<__main__.TrieNode object at 0x79b9478a1870>}, Συχνότητα: 45
Μοτίβο: {(0, 120.0): <__main__.TrieNode object at 0x79b947862bf0>, (1, 68.0):
<__main__.TrieNode object at 0x79b9478b1090>}, Συχνότητα: 62
Μοτίβο: {(0, 120.0): <__main__.TrieNode object at 0x79b947862bf0>, (1, 78.0):
<__main__.TrieNode object at 0x79b9478b3a00>}, Συχνότητα: 30
Μοτίβο: {(0, 120.0): <__main__.TrieNode object at 0x79b947862bf0>, (1, 75.0):
<__main__.TrieNode object at 0x79b9478b3dc0>}, Συχνότητα: 49
```

Ο κώδικας εντοπίζει μοτίβα που εμφανίζονται συχνά σύμφωνα με το κατώφλι υποστήριξης που ορίζεται στο 0.2%. Το κατώφλι υποστήριξης διαφοροποιείται καθώς για 2% δεν εμφανίζει αποτελέσματα. Στη δομή δεδομένων TRIE του αλγορίθμου καταγράφεται η συχνότητα εμφάνισης κάθε συνδυασμού και επιστρέφει αυτά που υπερβαίνουν το κατώφλι. Τα αποτελέσματα αυτά συνοψίζονται στον παρακάτω πίνακα.

Συστολική Πίεση	Διαστολική Πίεση	Αριθμός εμφάνισης
125.0	75.0	33
125.0	80.0	40
130.0	70.0	55
130.0	65.0	44
130.0	67.0	39

130.0	60.0	56
130.0	80.0	64
115.0	65.0	48
115.0	75.0	69
117.0	77.0	33
120.0	60.0	35
120.0	65.0	35
120.0	70.0	45
120.0	68.0	62
120.0	78.0	30
120.0	75.0	49

Πίνακας 4-3: Συχνά μοτίβα στα δεδομένα πίεσης

Οι τιμές της συστολικής πίεσης κυμαίνονται από 115.0 έως 130.0, με τις πιο συχνές τιμές να είναι γύρω από 120.0 και 130.0, που θεωρούνται φυσιολογικές. Οι τιμές της διαστολικής πίεσης κυμαίνονται από 60.0 έως 80.0, με τις πιο συχνές τιμές να εμφανίζονται γύρω από 65.0, 70.0 και 75.0. Και εδώ οι τιμές είναι εντός των φυσιολογικών ορίων. Οι πιο συχνοί συνδυασμοί είναι (115.0,75.0) με 69 εμφανίσεις και οι τιμές (130.0,80.0) με 64 εμφανίσεις είναι εντός του φυσιολογικού εύρους και θεωρούνται ένδειξη καλής υγείας.

Τα μοτίβα αυτά, μπορούν να χρησιμοποιηθούν για την εξαγωγή κανόνων συσχέτισης μεταξύ πίεσης και άλλων δεδομένων, όπως ο ύπνος, ή για την παρακολούθηση της υγείας της καρδιάς και της αρτηριακής πίεσης σε καθημερινή βάση.

ΚΕΦΑΛΑΙΟ 5

Επίλογος

5.1 Σκοπός έρευνας και γενικά συμπεράσματα

Σκοπός της συνολικής προετοιμασίας και επεξεργασίας, είναι η εξόρυξη γνώσης από τα δεδομένα υγείας συνεχούς ροής, με τον βέλτιστο δυνατό τρόπο. Διερευνάται δηλαδή η αποδοτικότερη επεξεργασία των δεδομένων προκειμένου να προκύψουν χρήσιμα και ορθά αποτελέσματα, χωρίς να χαθεί πολύτιμη πληροφορία, ενώ παράλληλα να μην οδηγηθεί ο αναλυτής σε υπερβολικές χρεώσεις, λόγω εκτεταμένης χρήσης μνήμης.

Η παρούσα διπλωματική ασχολήθηκε αρχικά με την προετοιμασία των δεδομένων, τον καθαρισμό τους δηλαδή από ακραίες τιμές και λάθη, την οπτικοποίησή τους σε γραφήματα και διαγράμματα, αλλά και την σύνοψη σε ιστογράμματα, ώστε να αποκτηθεί καλύτερη εικόνα. Στη συνέχεια πραγματεύτηκε μεθόδους συσταδοποίησης, κάνοντας χρήση του αλγορίθμου DBIECM και αναζητώντας το καλύτερο όριο για τις δημιουργούμενες συστάδες με τη χρήση του Silhouette Score. Τέλος συμπεριλήφθηκαν κανόνες συσχέτισης και συγκεκριμένα η εύρεση των συχνότερων μοτίβων.

Η εφαρμογή των παραπάνω οδήγησε σε χρήσιμα συμπεράσματα. Η οπτικοποίηση των δεδομένων βοήθησε στην καλύτερη κατανόησή τους. Χρησιμοποιώντας τα ιστογράμματα, αλλά και τον Local Outlier Factor, εντοπίσθηκαν τάσεις και ανωμαλίες (ακραίες τιμές) που δεν ήταν άμεσα προφανείς από τις καθαρές αριθμητικές τιμές. Η προ-επεξεργασία των δεδομένων αποκάλυψε επιπλέον προβλήματα ποιότητας δεδομένων, όπως ελλείποντα δεδομένα (στα δεδομένα βιοσημάτων) και σφάλματα καταγραφής (πολλαπλές καταγραφές του ίδιου συμπτώματος στα ερωτηματολόγια). Με τον τρόπο αυτό εξασφαλίστηκαν η ακρίβεια και η πληρότητα των δεδομένων. Από τα παραπάνω προέκυψαν επίσης αρκετά χρήσιμες παρατηρήσεις για τους χρήστες.

Όσον αφορά στη συσταδοποίηση και συγκεκριμένα στον αλγόριθμο DBIECM αναζητήθηκε η μέγιστη ακτίνα συστάδας χρησιμοποιώντας το Silhouette Score. Ένα κατάλληλο όριο διασφαλίζει ότι τα δεδομένα που ανήκουν στην ίδια κατηγορία είναι παρόμοια και σχετίζονται. Αν το όριο είναι πολύ χαμηλό ή υψηλό, μπορεί να προκύψουν εσφαλμένες ταξινομήσεις. Η συσταδοποίηση με τον αλγόριθμο DBIECM ωστόσο, δεν ήταν αρκετά αποτελεσματική, καθώς τα δεδομένα ήταν όμοια και υπήρχε επικάλυψη μεταξύ των συστάδων. Ενδεχομένως λοιπόν να έπρεπε να εφαρμοστεί ένας διαφορετικός αλγόριθμος. Τέλος στην εξόρυξη κανόνων συσχέτισης, αναζητήθηκαν τα πιο συχνά εμφανιζόμενα μοτίβα.

Εν κατακλείδι η εξόρυξη δεδομένων υγείας συνεχούς ροής είναι κρίσιμη για την έγκαιρη ανίχνευση και την αποτελεσματική παρακολούθηση της υγείας. Με την ανάλυση δεδομένων σε πραγματικό χρόνο από αισθητήρες και ιατρικές συσκευές, μπορούν να εντοπιστούν πρώιμα σημάδια παθολογικών καταστάσεων, να προσαρμοστούν θεραπείες ανάλογα με τις ανάγκες του ασθενούς, και να δημιουργηθούν ειδοποιήσεις για καταστάσεις που απαιτούν άμεση παρέμβαση. Η τεχνολογία αυτή βελτιώνει την ποιότητα της φροντίδας, μειώνει το κόστος υγειονομικών υπηρεσιών και ενισχύει την έρευνα και ανάπτυξη νέων θεραπευτικών μεθόδων. Συνολικά, η εξόρυξη δεδομένων υγείας συνεχούς ροής ενισχύει την αποτελεσματικότητα και την ασφάλεια των συστημάτων υγειονομικής περίθαλψης.

5.2 Προτάσεις για μελλοντική έρευνα

Η παρούσα Διπλωματική έχει προσπαθήσει να συνεισφέρει εφαρμόζοντας μεθόδους σχετικά με την εξόρυξη γνώσης από δεδομένα υγείας συνεχούς ροής. Ωστόσο, υπάρχουν αρκετές οδοί για μελλοντική έρευνα που μπορούν να εμπλουτίσουν περαιτέρω το πεδίο αυτό. Οι ακόλουθες προτάσεις προσφέρουν οδηγίες για μελλοντικές έρευνες:

1. Οπτικοποίηση και Αλληλεπίδραση Χρήστη: Η οπτικοποίηση δεδομένων και η αλληλεπίδραση χρήστη είναι κρίσιμες πτυχές της επεξεργασίας ροών δεδομένων υγείας, που επιτρέπουν στους παρόχους υγειονομικής περίθαλψης να κατανοούν και να αξιοποιούν αποτελεσματικά τα δεδομένα για τη λήψη αποφάσεων. Ορισμένες τεχνικές οπτικοποίησης για

ροές δεδομένων υγείας είναι οι διαδραστικοί πίνακες ελέγχου (dashboards) και η ανάλυση οπτικών δεδομένων σε πραγματικό χρόνο. Οι διαδραστικοί πίνακες ελέγχου παρέχουν μια συνολική δυναμική εικόνα των δεδομένων υγείας μέσω γραφημάτων, διαγραμμάτων και άλλων οπτικών στοιχείων, επιτρέποντας σε χρήστες να φιλτράρουν και να αναλύουν τα δεδομένα σε πραγματικό χρόνο. Η ανάλυση οπτικών δεδομένων σε πραγματικό χρόνο επιτρέπει την παρακολούθηση και ανάλυση δεδομένων καθώς αυτά ρέουν, προσφέροντας άμεσες πληροφορίες και ειδοποιήσεις. Η υλοποίησή τους μπορεί να γίνει μέσω Power Bi.

2. **Επιπλέον Τεχνικές Επεξεργασίας Δεδομένων:** Η μελλοντική έρευνα θα μπορούσε να επικεντρωθεί σε επιπλέον μεθόδους ανίχνευσης ακραίων τιμών όπως είναι ο Isolation Forest και οι Autoencoders. Ο αλγόριθμος Isolation Forest κατασκευάζει μία συλλογή από δέντρα απομόνωσης (δυαδικά δέντρα) με σκοπό να διαιρέσει τα δεδομένα. Στη συνέχεια υπολογίζεται το anomaly score για κάθε δεδομένο, ανάλογα με τη δυσκολία διαχωρισμού τους. Δεδομένα με χαμηλό anomaly score θεωρούνται ανωμαλίες ή ακραίες τιμές. (Zhiguo, Minrui, 2013) Οι autoencoders είναι μια κατηγορία αλγορίθμων μηχανικής μάθησης που χρησιμοποιούνται κυρίως για την εκμάθηση μιας συμπιεσμένης αναπαράστασης των δεδομένων. Στόχος τους είναι να ανακτήσουν τα αρχικά δεδομένα από μια συμπιεσμένη αναπαράσταση, παρέχοντας ένα ισχυρό εργαλείο για την ανάλυση, την ανακατασκευή και την ανίχνευση ανωμαλιών και ακραίων τιμών. Συγχρόνως θα μπορούσαν να ενσωματωθούν τεχνικές για την επεξεργασία πολυπαραγοντικών ροών δεδομένων, όπως η Ανάλυση Κύριων Συνιστωσών (PCA).

3. **Σύνθεση Δεδομένων:** Η σύνθεση δεδομένων από διαφορετικές πηγές είναι μια κρίσιμη διαδικασία σε πολλές εφαρμογές, ιδιαίτερα στον τομέα της υγείας και επομένως αποτελεί πρόσφορο έδαφος για επιπλέον έρευνα. Η σύνθεση αυτή περιλαμβάνει την ενοποίηση και ανάλυση δεδομένων που προέρχονται από διαφορετικές πηγές για την εξαγωγή χρήσιμων πληροφοριών και την ενίσχυση της λήψης αποφάσεων. Η ενοποίηση των δεδομένων μπορεί να πραγματοποιηθεί με συγχώνευση των δεδομένων (πχ ενοποίηση πινάκων ή συγχώνευση εγγραφών με κοινά πεδία) και με εργαλεία ETL (Extract, Transform, Load), όπως είναι το SSIS.

4. **Ανάλυση σε Πραγματικό Χρόνο και Υποστήριξη Αποφάσεων:** Η διερεύνηση τρόπων με τους οποίους τα συστήματα επεξεργασίας ροών μπορούν να ενσωματωθούν σε συστήματα

υποστήριξης αποφάσεων για την παροχή ειδοποιήσεων και συστάσεων σε πραγματικό χρόνο, θα μπορούσε να είναι μία πολλά υποσχόμενη κατεύθυνση. Για παράδειγμα σε μονάδες εντατικής θεραπείας, συστήματα επεξεργασίας ροών μπορούν να παρακολουθούν ζωτικά σημεία ασθενών και να ειδοποιούν το ιατρικό προσωπικό για ενδεχόμενες επικίνδυνες καταστάσεις. Επιπλέον χρήσιμο θα ήταν να διερευνηθούν τεχνικές πρόβλεψης όπως πρόβλεψη χρονοσειρών (ARIMA: που λαμβάνει υπόψιν την τάση, την εποχικότητα και τις διακυμάνσεις των δεδομένων) ή μοντέλα μηχανικής μάθησης (π.χ. παλινδρόμηση, ταξινόμηση) για την πρόβλεψη γεγονότων υγείας.

5. Αναδυόμενες τεχνολογίες: Οι αναδυόμενες τεχνολογίες παίζουν καθοριστικό ρόλο στην εξέλιξη της επεξεργασίας ροών δεδομένων υγείας. Χαρακτηριστικό παράδειγμα αποτελεί το Blockchain που προσφέρει μια ασφαλή και αμετάβλητη πλατφόρμα για την αποθήκευση και διαχείριση των δεδομένων υγείας. Η τεχνολογία αυτή μπορεί να χρησιμοποιηθεί για την διασφάλιση της αυθεντικότητας και της ακεραιότητας των δεδομένων, καθώς και για τη διευκόλυνση της διαμοιρασμένης πρόσβασης. (Aitizaz, Hashim, Aamir, Aftab, Ting, Muhammad, Yazeed ,Heba, 2023)

6. Δείκτες Αξιολόγησης: Η αξιολόγηση των τεχνικών επεξεργασίας ροών δεδομένων υγείας είναι κρίσιμη για τη διασφάλιση της αποτελεσματικότητας και της αποδοτικότητας αυτών των συστημάτων. Οι δείκτες απόδοσης και οι μέθοδοι αξιολόγησης παρέχουν τις απαραίτητες πληροφορίες για τη βελτίωση και την προσαρμογή των τεχνικών αυτών, ώστε να ικανοποιούν τις ανάγκες των χρηστών και να ανταποκρίνονται στις απαιτήσεις του τομέα της υγείας. Τέτοιοι δείκτες είναι οι ακόλουθοι:

- **Η ακρίβεια** αποτελεί το ποσοστό των σωστών προβλέψεων.
- **Η καθυστέρηση** είναι ο χρόνος που απαιτείται για την επεξεργασία και την παροχή αποτελέσματος.
- **Ο ρυθμός μετάδοσης** είναι το ποσό των δεδομένων που μπορεί να επεξεργαστεί ένα σύστημα σε μια συγκεκριμένη χρονική περίοδο.
- **Η κλιμάκωση** είναι η ικανότητα του συστήματος να διατηρεί την απόδοση του ενώ αυξάνονται τα δεδομένα.

- **Η ευρωστία** είναι η ικανότητα του συστήματος να διατηρεί την απόδοσή του σε περίπτωση σφαλμάτων. (Aitizaz, Hashim, Aamir , Aftab, Ting, Muhammad, Yazeed, Heba,2023)

7. Ιδιωτικότητα και Ασφάλεια: Η διασφάλιση της ιδιωτικότητας των δεδομένων υγείας είναι ιδιαίτερα σημαντική λόγω της ευαίσθητης φύσης τους. Δύο τεχνικές για τη διασφάλιση της ιδιωτικότητας των δεδομένων σε ροές δεδομένων υγείας αποτελούν η διαφορική ιδιωτικότητα και ανωνυμοποίηση. Η διαφορική ιδιωτικότητα είναι μια τεχνική που επιτρέπει την εξαγωγή συνολικών πληροφοριών από ένα σύνολο δεδομένων χωρίς να αποκαλύπτει τα δεδομένα των ατόμων. Οι τεχνικές ανωνυμοποίησης περιλαμβάνουν μεθόδους όπως η αφαίρεση ή η γενίκευση αναγνωριστικών πληροφοριών (όπως ονόματα ή διευθύνσεις) και η χρήση ψευδώνυμων για την αντικατάσταση ευαίσθητων δεδομένων. Επιπλέον μπορεί να γίνει κρυπτογράφηση των δεδομένων.

8. Ηθικές και Ρυθμιστικές Εκτιμήσεις: Οι ηθικές και ρυθμιστικές εκτιμήσεις παίζουν κρίσιμο ρόλο στη συλλογή, επεξεργασία και χρήση δεδομένων υγείας. Αυτές οι εκτιμήσεις εξασφαλίζουν ότι οι πρακτικές διαχείρισης δεδομένων δεν παραβιάζουν τα δικαιώματα των ασθενών όπως η συναίνεση, η ιδιοκτησία δεδομένων και η επίδραση των αλγοριθμικών αποφάσεων στην φροντίδα τους και συμμορφώνονται με τους ισχύοντες νόμους και κανονισμούς. Παράλληλα έχει ενδιαφέρον να εξεταστεί το ρυθμιστικό περιβάλλον για την επεξεργασία δεδομένων υγείας, συμπεριλαμβανομένης της συμμόρφωσης με κανονισμούς όπως το GDPR (General Data Protection Regulation) που εφαρμόζεται στην Ευρωπαϊκή Ένωση και προβλέπει τη διασφάλιση της ιδιωτικότητας των δεδομένων, τη λήψη συγκατάθεσης από τους υποκείμενους των δεδομένων στους οποίους δίνεται και δικαίωμα επεξεργασίας αλλά και διαγραφής των δεδομένων, την περιορισμένη χρήση των δεδομένων, μόνο για σκοπό που έχουν δώσει συναίνεση, την ειδοποίηση της αρμόδιας εποπτικής αρχής σε περίπτωση παραβίασης των δεδομένων και την εκτίμηση των πιθανών κινδύνων και επιπτώσεων των δραστηριοτήτων επεξεργασίας δεδομένων (Delev, 2024).

ΠΑΡΑΡΤΗΜΑ

Κώδικας υλοποίησης της παρούσας Διπλωματικής

```
# prompt: open sleep.xlsx
import pandas as pd
# Open the Excel file and read the "sleep" sheet
df = pd.read_excel("/sleep.xlsx")
# Print the DataFrame
print(df)
# prompt: print column deep
rem=(df['Column1._source.rem'])
light=(df['Column1._source.light'])
total=(df['Column1._source.total'])
deep=(df['Column1._source.deep'])
awake=(df['Column1._source.awake'])
print(deep)
print(awake)
print(total)
print(rem)
print(light)
percrem=rem/total
perclight=light/total
print(percrem)
print(perclight)
# prompt: open steps.xlsx
import pandas as pd
# Open the Excel file and read the "steps" sheet
df = pd.read_excel("/steps.xlsx")
# Print the DataFrame
print(df)
# prompt: print column deep
steps=(df['Column1._source.steps'])
calories=(df['Column1._source.calories'])
distance=(df['Column1._source.distance'])
print(steps)
print(calories)

# prompt: open measurements.xlsx
import pandas as pd
# Open the Excel file and read the "measurements" sheet
df = pd.read_excel("/measurements.xlsx")
# Print the DataFrame
print(df)
# prompt: print column deep
```

```
systolic=(df['Column1._source.systolic.value'])
diastolic=(df['Column1._source.diastolic.value'])
systolic.dropna(inplace=True)
diastolic.dropna(inplace=True)
print(systolic)
print(diastolic)
heartrate=(df['Column1._source.heartrate.value'])
heartrate.dropna(inplace=True)
print(heartrate)
weight=(df['Column1._source.weight.value'])
weight.dropna(inplace=True)
print(weight)
respiratory=(df['Column1._source.respiratory_rate.value'])
respiratory.dropna(inplace=True)
print(respiratory)
temperature=(df['Column1._source.body_temperature.value'])
temperature.dropna(inplace=True)
print(temperature)
spo2=(df['Column1._source.spo2.value'])
spo2.dropna(inplace=True)
print(spo2)

mean_steps=steps.mean()
mean_calories=calories.mean()
mean_distance=distance.mean()
print(mean_steps)
print(mean_calories)
print(mean_distance)
mean_systolic=systolic.mean()
mean_diastolic=diastolic.mean()
mean_heartrate=heartrate.mean()
mean_respiratory=respiratory.mean()
mean_temperature=temperature.mean()
mean_spo2=spo2.mean()
mean_weight=weight.mean()
print(mean_systolic)
print(mean_diastolic)
print(mean_heartrate)
print(mean_respiratory)
print(mean_temperature)
print(mean_spo2)
print(mean_weight)
mean_light=light.mean()
mean_rem=rem.mean()
mean_deep=deep.mean()
mean_awake=awake.mean()
```

```
mean_total=total.mean()
print(mean_light)
print(mean_rem)
print(mean_deep)
print(mean_awake)
print(mean_total)
variancesteps=steps.var()
variancecalories=calories.var()
variancedistance=distance.var()
print(variancesteps)
print(variancecalories)
print(variancedistance)
varianceawake=awake.var()
variancedeep=deep.var()
variancelight=light.var()
variancetotal=total.var()
variancerem=rem.var()
print(varianceawake)
print(variancedeep)
print(variancelight)
print(variancetotal)
print(variancerem)
variancediastolic=diastolic.var()
variancesystolic=systolic.var()
varianceheartrate=heartrate.var()
variancetemperature=temperature.var()
varianceweight=weight.var()
variancepo2=spo2.var()
varianceresiratory=respiratory.var()
print(variancediastolic)
print(variancesystolic)
print(varianceheartrate)
print(variancetemperature)
print(varianceweight)
print(variancepo2)
print(varianceresiratory)
minsteps=steps.min()
maxsteps=steps.max()
mincalories=calories.min()
maxcalories=calories.max()
mindis=distance.min()
maxdis=distance.max()
minlight=light.min()
maxlight=light.max()
mindeep=deep.min()
maxdeep=deep.max()
```

```
minawake=awake.min()
maxawake=awake.max()
mintotal=total.min()
maxtotal=total.max()
minrem=rem.min()
maxrem=rem.max()
mindiastolic=diastolic.min()
maxdiastolic=diastolic.max()
minsypstolic=systolic.min()
maxsystolic=systolic.max()
minheartrate=heartrate.min()
maxheartrate=heartrate.max()
mintemperature=temperature.min()
maxtemperature=temperature.max()
minweight=weight.min()
maxweight=weight.max()
minpo2=spo2.min()
maxpo2=spo2.max()
minrespiratory=respiratory.min()
maxrespiratory=respiratory.max()
print(minsteps)
print(maxsteps)
print(mincalories)
print(maxcalories)
print(mindis)
print(maxdis)
print(minlight)
print(maxlight)
print(mindeep)
print(maxdeep)
print(minawake)
print(maxawake)
print(mintotal)
print(maxtotal)
print(minrem)
print(maxrem)
print(mindiastolic)
print(maxdiastolic)
print(minsypstolic)
print(maxsystolic)
print(minheartrate)
print(maxheartrate)
print(mintemperature)
print(maxtemperature)
print(minweight)
print(maxweight)
```

```
print(minpo2)
print(maxpo2)
print(minrespiratory)
print(maxrespiratory)

# Fit Local Outlier Factor model
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt
import pandas as pd
clf = LocalOutlierFactor(n_neighbors=100)
charges = pd.DataFrame({'DEEP': deep, 'LIGHT': light})
clf.fit_predict(charges)  # Fit the model
charges['LOF_Score'] = clf.negative_outlier_factor_
# Calculate PCT column (optional, based on your needs)
charges['PCT'] = charges['DEEP'] / charges['LIGHT']
# Plotting
plt.figure(figsize=(10, 6))
# Scatter plot for normal points
plt.scatter(charges['DEEP'], charges['LIGHT'], c='black', s=50,
label="Data Points")
# Radius for outliers
radius = (charges['LOF_Score'].max() - charges['LOF_Score']) /
(charges['LOF_Score'].max() - charges['LOF_Score'].min())
plt.scatter(charges['REM'], charges['DEEP'], s=250 * radius,
edgecolors="r", facecolors="none", label="Outlier scores")
# Adjust legend sizes
legend = plt.legend(loc="upper left")
legend.legendHandles[0]._sizes = [100]
legend.legendHandles[1]._sizes = [100]
plt.xlabel('DEEP')
plt.ylabel('LIGHT')
plt.title('Outlier Detection using Local Outlier Factor')
plt.show()
# Print the dataset with LOF scores
print(charges)


# Fit Local Outlier Factor model
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt
import pandas as pd
clf = LocalOutlierFactor(n_neighbors=100)
charges = pd.DataFrame({'REM': rem, 'LIGHT': light})
clf.fit_predict(charges)  # Fit the model
charges['LOF_Score'] = clf.negative_outlier_factor_
# Calculate PCT column (optional, based on your needs)
```

```
charges['PCT'] = charges['REM'] / charges['LIGHT']
# Plotting
plt.figure(figsize=(10, 6))
# Scatter plot for normal points
plt.scatter(charges['REM'], charges['LIGHT'], c='khaki', s=50, label="Data
Points")
# Radius for outliers
radius = (charges['LOF_Score'].max() - charges['LOF_Score']) /
(charges['LOF_Score'].max() - charges['LOF_Score'].min())
plt.scatter(charges['REM'], charges['DEEP'], s=230 * radius,
edgecolors="r", facecolors="none", label="Outlier scores")
# Adjust legend sizes
legend = plt.legend(loc="upper left")
legend.legendHandles[0]._sizes = [100]
legend.legendHandles[1]._sizes = [100]
plt.xlabel('REM')
plt.ylabel('LIGHT')
plt.title('Outlier Detection using Local Outlier Factor')
plt.show()
# Print the dataset with LOF scores
print(charges)


# Fit Local Outlier Factor model
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt
import pandas as pd
clf = LocalOutlierFactor(n_neighbors=100)
charges = pd.DataFrame({'REM': rem, 'DEEP': deep})
clf.fit_predict(charges)  # Fit the model
charges['LOF_Score'] = clf.negative_outlier_factor_
# Calculate PCT column (optional, based on your needs)
charges['PCT'] = charges['REM'] / charges['DEEP']
# Plotting
plt.figure(figsize=(10, 6))
# Scatter plot for normal points
plt.scatter(charges['REM'], charges['DEEP'], c='orange', s=50, label="Data
Points")
# Radius for outliers
radius = (charges['LOF_Score'].max() - charges['LOF_Score']) /
(charges['LOF_Score'].max() - charges['LOF_Score'].min())
plt.scatter(charges['REM'], charges['DEEP'], s=150 * radius,
edgecolors="r", facecolors="none", label="Outlier scores")
# Adjust legend sizes
legend = plt.legend(loc="upper left")
legend.legendHandles[0]._sizes = [100]
```

```
legend.legendHandles[1]._sizes = [100]
plt.xlabel('REM')
plt.ylabel('DEEP')
plt.title('Outlier Detection using Local Outlier Factor')
plt.show()
# Print the dataset with LOF scores
print(charges)


# Fit Local Outlier Factor model
clf = LocalOutlierFactor(n_neighbors=30)
charges = pd.DataFrame({'STEPS': steps, 'CALORIES': calories})
clf.fit_predict(charges)  # Fit the model
charges['LOF_Score'] = clf.negative_outlier_factor_
# Calculate PCT column (optional, based on your needs)
charges['PCT'] = charges['STEPS'] / charges['CALORIES']
# Plotting
plt.figure(figsize=(10, 6))
# Scatter plot for normal points
plt.scatter(charges['STEPS'], charges['CALORIES'], c='green', s=50,
label="Data Points")
# Radius for outliers
radius = (charges['LOF_Score'].max() - charges['LOF_Score']) /
(charges['LOF_Score'].max() - charges['LOF_Score'].min())
plt.scatter(charges['STEPS'], charges['CALORIES'], s=200 * radius,
edgecolors="r", facecolors="none", label="Outlier scores")
# Adjust legend sizes
legend = plt.legend(loc="upper left")
legend.legendHandles[0]._sizes = [100]
legend.legendHandles[1]._sizes = [100]
plt.xlabel('STEPS')
plt.ylabel('CALORIES')
plt.title('Outlier Detection using Local Outlier Factor')
plt.show()
# Print the dataset with LOF scores
print(charges)

# Fit Local Outlier Factor model
clf = LocalOutlierFactor(n_neighbors=30)
charges = pd.DataFrame({'SYSTOLIC': systolic, 'DIASTOLIC': diastolic})
clf.fit_predict(charges)  # Fit the model
charges['LOF_Score'] = clf.negative_outlier_factor_
# Calculate PCT column (optional, based on your needs)
charges['PCT'] = charges['SYSTOLIC'] / charges['DIASTOLIC']
# Plotting
plt.figure(figsize=(10, 6))
```

```
# Scatter plot for normal points
plt.scatter(charges['SYSTOLIC'], charges['DIASTOLIC'], c='purple', s=50,
label="Data Points")
# Radius for outliers
radius = (charges['LOF_Score'].max() - charges['LOF_Score']) /
(charges['LOF_Score'].max() - charges['LOF_Score'].min())
plt.scatter(charges['SYSTOLIC'], charges['DIASTOLIC'], s=230 * radius,
edgecolors="r", facecolors="none", label="Outlier scores")
# Adjust legend sizes
legend = plt.legend(loc="upper left")
legend.legendHandles[0]._sizes = [100]
legend.legendHandles[1]._sizes = [100]
plt.xlabel('SYSTOLIC')
plt.ylabel('DIASTOLIC')
plt.title('Outlier Detection using Local Outlier Factor')
plt.show()
# Print the dataset with LOF scores
print(charges)


#Decayed histogram for Total Sleep
df = pd.DataFrame({'Total Sleep': total})
# Step 1: Apply exponential weights
alpha = 0.1  # Decay rate
# Compute exponentially weighted moving average
df['ewm'] = df['Total Sleep'].ewm(alpha=alpha, adjust=False).mean() # Use
'Total Sleep' instead of 'total'
# Step 2: Plot the decayed histogram with custom colors
plt.figure(figsize=(10, 8))
# Plot original data histogram
plt.hist(df['Total Sleep'], bins=10, alpha=0.5, label='Original Data',
edgecolor='black', color='pink') # Use 'Total Sleep' instead of 'total'
# Plot decayed data histogram
plt.hist(df['ewm'], bins=10, alpha=0.5, label='Decayed Data',
edgecolor='black', color='violet')
# Add titles and labels
plt.title('Decayes Histogram of Total Sleep')
plt.xlabel('Total Sleep Value')
plt.ylabel('Frequency')
plt.legend()
# Show the plot
plt.show()

#Decayed Histogram of Respiratory Data
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.DataFrame({'respiratory': respiratory})
# Step 1: Apply exponential weights
alpha = 0.1  # Decay rate
# Compute exponentially weighted moving average
df['ewm'] = df['respiratory'].ewm(alpha=alpha, adjust=False).mean()
# Step 2: Plot the decayed histogram
plt.figure(figsize=(10, 6))
# Plot original data histogram
plt.hist(df['respiratory'], bins=10, alpha=0.5, label='Original Data',
edgecolor='black')
# Plot decayed data histogram
plt.hist(df['ewm'], bins=10, alpha=0.5, label='Decayed Data',
edgecolor='black')
# Add titles and labels
plt.title('Decayed Histogram of Respiratory Data')
plt.xlabel('Respiratory Value')
plt.ylabel('Frequency')
plt.legend()
# Show the plot
plt.show()

#Decayed Histogram for Heartrate
df = pd.DataFrame({'heartrate': heartrate})
# Step 1: Apply exponential weights
alpha = 0.1  # Decay rate
# Compute exponentially weighted moving average
df['ewm'] = df['heartrate'].ewm(alpha=alpha, adjust=False).mean()
# Step 2: Plot the decayed histogram with custom colors
plt.figure(figsize=(10, 6))
# Plot original data histogram
plt.hist(df['heartrate'], bins=10, alpha=0.5, label='Original Data',
edgecolor='black', color='green')
# Plot decayed data histogram
plt.hist(df['ewm'], bins=10, alpha=0.5, label='Decayed Data',
edgecolor='black', color='pink')
# Add titles and labels
plt.title('Decayed Histogram of Heartrate Data')
plt.xlabel('Heartrate Value')
plt.ylabel('Frequency')
plt.legend()
# Show the plot
plt.show()

#Decayed Histogram for SPO2
df = pd.DataFrame({'spo2': spo2})
# Step 1: Apply exponential weights
```

```
alpha = 0.1  # Decay rate
# Compute exponentially weighted moving average
df['ewm'] = df['spo2'].ewm(alpha=alpha, adjust=False).mean()
# Step 2: Plot the decayed histogram with custom colors
plt.figure(figsize=(10, 8))
# Plot original data histogram
plt.hist(df['spo2'], bins=10, alpha=0.5, label='Original Data',
edgecolor='black', color='red')
# Plot decayed data histogram
plt.hist(df['ewm'], bins=10, alpha=0.5, label='Decayed Data',
edgecolor='black', color='purple')
# Add titles and labels
plt.title('Decayes Histogram of SPO2 Data')
plt.xlabel('SPO2 Value')
plt.ylabel('Frequency')
plt.legend()
# Set axis limits
plt.xlim(70, 110)  # Set x-axis limit
# Show the plot
plt.show()

#Decayed histogram for Body Temperature
df = pd.DataFrame({'temperature': temperature})
# Step 1: Apply exponential weights
alpha = 0.1  # Decay rate
# Compute exponentially weighted moving average
df['ewm'] = df['temperature'].ewm(alpha=alpha, adjust=False).mean()
# Step 2: Plot the decayed histogram with custom colors
plt.figure(figsize=(10, 8))
# Plot original data histogram
plt.hist(df['temperature'], bins=10, alpha=0.5, label='Original Data',
edgecolor='black', color='orange')
# Plot decayed data histogram
plt.hist(df['ewm'], bins=10, alpha=0.5, label='Decayed Data',
edgecolor='black', color='yellow')
# Add titles and labels
plt.title('Decayes Histogram of Body Temperature Data')
plt.xlabel('Body Temperature Value')
plt.ylabel('Frequency')
plt.legend()
# Show the plot
plt.show()

#Decayed histogram for Weight
df = pd.DataFrame({'weight': weight})
# Step 1: Apply exponential weights
```

```
alpha = 0.1  # Decay rate
# Compute exponentially weighted moving average
df['ewm'] = df['weight'].ewm(alpha=alpha, adjust=False).mean()
# Step 2: Plot the decayed histogram with custom colors
plt.figure(figsize=(10, 8))
# Plot original data histogram
plt.hist(df['weight'], bins=10, alpha=0.5, label='Original Data',
edgecolor='black', color='brown')
# Plot decayed data histogram
plt.hist(df['ewm'], bins=10, alpha=0.5, label='Decayed Data',
edgecolor='black', color='red')
# Add titles and labels
plt.title('Decayes Histogram of Weight Data')
plt.xlabel('Weight Value')
plt.ylabel('Frequency')
plt.legend()
# Show the plot
plt.show()

df = pd.DataFrame({'Total Sleep': total })
# Step 1: Define the window size and step size
window_size = 100
step_size = 50
windows_per_plot = 5  # Number of windows to display per plot
# Step 2: Process the data in overlapping windows
def windowed_histogram(df, window_size, step_size, windows_per_plot):
    num_windows = (len(df) - window_size) // step_size + 1
    num_plots = (num_windows + windows_per_plot - 1) // windows_per_plot
    for plot_index in range(num_plots):
        fig, axes = plt.subplots(min(windows_per_plot, num_windows -
plot_index * windows_per_plot), 1, figsize=(10, 2 * windows_per_plot),
constrained_layout=True)
        if min(windows_per_plot, num_windows - plot_index *
windows_per_plot) == 1:
            axes = [axes]  # Ensure axes is a list even if there's only one
plot
        for i in range(windows_per_plot):
            window_index = plot_index * windows_per_plot + i
            if window_index >= num_windows:
                break
            start = window_index * step_size
            end = start + window_size
            window_data = df['Total Sleep'][start:end]

            ax = axes[i]
```

```
            ax.hist(window_data, bins=10, edgecolor='black',
color='#1f77b4', alpha=0.7)
            ax.set_title(f'Window {window_index + 1}: {start} to {end}')
            ax.set_xlabel('Total Sleep')
            ax.set_ylabel('Frequency')
        plt.show()
# Step 3: Plot histograms for each window
windowed_histogram(df, window_size, step_size, windows_per_plot)



df = pd.DataFrame({'respiratory': respiratory})
# Step 1: Define the window size and step size
window_size = 1000
step_size = 500
windows_per_plot = 5  # Number of windows to display per plot
# Step 2: Process the data in overlapping windows
def windowed_histogram(df, window_size, step_size, windows_per_plot):
    num_windows = (len(df) - window_size) // step_size + 1
    num_plots = (num_windows + windows_per_plot - 1) // windows_per_plot
    for plot_index in range(num_plots):
        fig, axes = plt.subplots(min(windows_per_plot, num_windows -
plot_index * windows_per_plot), 1, figsize=(10, 2 * windows_per_plot),
constrained_layout=True)
        if min(windows_per_plot, num_windows - plot_index *
windows_per_plot) == 1:
            axes = [axes]  # Ensure axes is a list even if there's only one
plot
        for i in range(windows_per_plot):
            window_index = plot_index * windows_per_plot + i
            if window_index >= num_windows:
                break
            start = window_index * step_size
            end = start + window_size
            window_data = df['respiratory'][start:end]

            ax = axes[i]
            ax.hist(window_data, bins=10, edgecolor='black',
color='#1f77b4', alpha=0.7)
            ax.set_title(f'Window {window_index + 1}: {start} to {end}')
            ax.set_xlabel('Respiratory Value')
            ax.set_ylabel('Frequency')
        plt.show()
# Step 3: Plot histograms for each window
windowed_histogram(df, window_size, step_size, windows_per_plot)
```

```
df = pd.DataFrame({'heartrate': heartrate})
# Step 1: Define the window size and step size
window_size = 30000
step_size = 25000
windows_per_plot = 10  # Number of windows to display per plot
# Step 2: Process the data in overlapping windows
def windowed_histogram(df, window_size, step_size, windows_per_plot):
    num_windows = (len(df) - window_size) // step_size + 1
    num_plots = (num_windows + windows_per_plot - 1) // windows_per_plot
    for plot_index in range(num_plots):
        fig, axes = plt.subplots(min(windows_per_plot, num_windows -
plot_index * windows_per_plot), 1, figsize=(10, 2 * windows_per_plot),
constrained_layout=True)
        if min(windows_per_plot, num_windows - plot_index *
windows_per_plot) == 1:
            axes = [axes]  # Ensure axes is a list even if there's only one
plot
        for i in range(windows_per_plot):
            window_index = plot_index * windows_per_plot + i
            if window_index >= num_windows:
                break
            start = window_index * step_size
            end = start + window_size
            window_data = df['heartrate'][start:end]

            ax = axes[i]
            ax.hist(window_data, bins=10, edgecolor='black',
color='#1f77b4', alpha=0.7)
            ax.set_title(f'Window {window_index + 1}: {start} to {end}')
            ax.set_xlabel('heartrate Value')
            ax.set_ylabel('Frequency')
        plt.show()
# Step 3: Plot histograms for each window
windowed_histogram(df, window_size, step_size, windows_per_plot)

df = pd.DataFrame({'spo2': spo2})
# Step 1: Define the window size and step size
window_size = 4000
step_size = 2000
windows_per_plot = 10  # Number of windows to display per plot
# Step 2: Process the data in overlapping windows
def windowed_histogram(df, window_size, step_size, windows_per_plot):
    num_windows = (len(df) - window_size) // step_size + 1
    num_plots = (num_windows + windows_per_plot - 1) // windows_per_plot
    for plot_index in range(num_plots):
```

```
        fig, axes = plt.subplots(min(windows_per_plot, num_windows -
plot_index * windows_per_plot), 1, figsize=(10, 2 * windows_per_plot),
constrained_layout=True)
        if min(windows_per_plot, num_windows - plot_index *
windows_per_plot) == 1:
            axes = [axes]  # Ensure axes is a list even if there's only one
plot
        for i in range(windows_per_plot):
            window_index = plot_index * windows_per_plot + i
            if window_index >= num_windows:
                break
            start = window_index * step_size
            end = start + window_size
            window_data = df['spo2'][start:end]
            ax = axes[i]
            ax.hist(window_data, bins=10, edgecolor='black',
color='#1f77b4', alpha=0.7)
            ax.set_title(f'Window {window_index + 1}: {start} to {end}')
            ax.set_xlabel('spo2 Value')
            ax.set_ylabel('Frequency')
        plt.show()
# Step 3: Plot histograms for each window
windowed_histogram(df, window_size, step_size, windows_per_plot)

df = pd.DataFrame({'temperature': temperature})
# Step 1: Define the window size and step size
window_size = 1000
step_size = 500
windows_per_plot = 10  # Number of windows to display per plot
# Step 2: Process the data in overlapping windows
def windowed_histogram(df, window_size, step_size, windows_per_plot):
    num_windows = (len(df) - window_size) // step_size + 1
    num_plots = (num_windows + windows_per_plot - 1) // windows_per_plot
    for plot_index in range(num_plots):
        fig, axes = plt.subplots(min(windows_per_plot, num_windows -
plot_index * windows_per_plot), 1, figsize=(10, 2 * windows_per_plot),
constrained_layout=True)

        if min(windows_per_plot, num_windows - plot_index *
windows_per_plot) == 1:
            axes = [axes]  # Ensure axes is a list even if there's only one
plot
        for i in range(windows_per_plot):
            window_index = plot_index * windows_per_plot + i
            if window_index >= num_windows:
                break
```

```
            start = window_index * step_size
            end = start + window_size
            window_data = df['temperature'][start:end]
            ax = axes[i]
            ax.hist(window_data, bins=10, edgecolor='black',
color='#1f77b4', alpha=0.7)
            ax.set_title(f'Window {window_index + 1}: {start} to {end}')
            ax.set_xlabel('temperature Value')
            ax.set_ylabel('Frequency')
        plt.show()
# Step 3: Plot histograms for each window
windowed_histogram(df, window_size, step_size, windows_per_plot)


df = pd.DataFrame({'weight': weight})
# Step 1: Define the window size and step size
window_size = 1000
step_size = 500
windows_per_plot = 10  # Number of windows to display per plot
# Step 2: Process the data in overlapping windows
def windowed_histogram(df, window_size, step_size, windows_per_plot):
    num_windows = (len(df) - window_size) // step_size + 1
    num_plots = (num_windows + windows_per_plot - 1) // windows_per_plot
    for plot_index in range(num_plots):
        fig, axes = plt.subplots(min(windows_per_plot, num_windows -
plot_index * windows_per_plot), 1, figsize=(10, 2 * windows_per_plot),
constrained_layout=True)
        if min(windows_per_plot, num_windows - plot_index *
windows_per_plot) == 1:
            axes = [axes]  # Ensure axes is a list even if there's only one
plot
        for i in range(windows_per_plot):
            window_index = plot_index * windows_per_plot + i
            if window_index >= num_windows:
                break
            start = window_index * step_size
            end = start + window_size
            window_data = df['weight'][start:end]

            ax = axes[i]
            ax.hist(window_data, bins=10, edgecolor='black',
color='#1f77b4', alpha=0.7)
            ax.set_title(f'Window {window_index + 1}: {start} to {end}')
            ax.set_xlabel('Weight Value')
            ax.set_ylabel('Frequency')
        plt.show()
```

```
# Step 3: Plot histograms for each window
windowed_histogram(df, window_size, step_size, windows_per_plot)

from mpl_toolkits.mplot3d import Axes3D
import numpy as np # Import numpy
import matplotlib.pyplot as plt # Import pyplot for plotting
# Distance calculation function
def distance(point1, point2):
    return np.linalg.norm(point1 - point2)
# Update cluster center function
def update_cluster_center(cluster):
    return np.mean(cluster, axis=0)
# Update cluster radius function
def update_cluster_radius(cluster, center):
    return max([distance(point, center) for point in cluster])
# DBIECM
def dbiecm(data, D_thr):
    clusters = []
    cluster_centers = []
    cluster_radii = []
    for x_i in data:
        if not clusters:  #Create the first cluster
            clusters.append([x_i])
            cluster_centers.append(x_i)
            cluster_radii.append(0)
        else:
            distances = [distance(x_i, center) for center in
cluster_centers]
            min_distance = min(distances)
            min_index = distances.index(min_distance)
            if min_distance > D_thr:
                # Create new cluster
                clusters.append([x_i])
                cluster_centers.append(x_i)
                cluster_radii.append(0)
            else:
                # Add to an existing cluster
                clusters[min_index].append(x_i)
                cluster_centers[min_index] =
update_cluster_center(clusters[min_index])
                cluster_radii[min_index] =
update_cluster_radius(clusters[min_index], cluster_centers[min_index])
    return clusters, cluster_centers, cluster_radii
data_array = df[['Column1._source.calories', 'Column1._source.distance',
'Column1._source.steps']].values
# Set Dthr
D_thr = 3000
```

```
# Apply DBIECM
clusters, centers, radii = dbiecm(data_array, D_thr)
for i, (cluster, center, radius) in enumerate(zip(clusters, centers,
radii)):
    print(f"Cluster {i+1}:")
    print(f"  Points: {cluster}")
    print(f"  Center: {center}")
    print(f"  Radius: {radius}")
# Create diagrams
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
colors = ['r', 'g', 'b', 'y', 'c', 'm', 'k']
for i, cluster in enumerate(clusters):
    cluster = np.array(cluster)
    ax.scatter(cluster[:, 0], cluster[:, 1], cluster[:, 2], c=colors[i %
len(colors)], label=f'Cluster {i+1}')
    center = centers[i]
    radius = radii[i]
    ax.scatter(center[0], center[1], center[2], c='k', marker='x')
    # Plot the radius as a sphere
    u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
    x = center[0] + radius * np.cos(u) * np.sin(v)
    y = center[1] + radius * np.sin(u) * np.sin(v)
    z = center[2] + radius * np.cos(v)
    ax.plot_wireframe(x, y, z, color=colors[i % len(colors)], alpha=0.3)
ax.set_xlabel('Calories')
ax.set_ylabel('Distance')
ax.set_zlabel('Steps')
ax.legend()
plt.show()

import numpy as np
import pandas as pd
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
# Load and preprocess data
X =
df[['Column1._source.calories','Column1._source.distance','Column1._sour
ce.steps']].values
X = StandardScaler().fit_transform(X)  # Normalize data
# Define a range of D_thr values to test
D_thr_values = np.linspace(1, 100, 500)  # Example range; adjust based
on data scale
best_D_thr = None
best_score = -1
```

```
scores = []
valid_D_thr_values = [] # Store D_thr values for which silhouette score
is calculated
for D_thr in D_thr_values:
    # Run DBIECM
    clusters, cluster_centers, cluster_radii = dbiecm(X, D_thr)
    # Flatten clusters for silhouette score calculation
    labels = np.concatenate([np.full(len(cluster), i) for i, cluster in
enumerate(clusters)])
    # Compute silhouette score
    if len(clusters) > 1:  # Silhouette score requires at least 2
clusters
        score = silhouette_score(X, labels)
        scores.append(score)
        valid_D_thr_values.append(D_thr) # Store the corresponding D_thr
value
        if score > best_score:
            best_score = score
            best_D_thr = D_thr
# Plot silhouette scores using valid_D_thr_values
plt.plot(valid_D_thr_values, scores)
plt.xlabel('D_thr')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score vs D_thr')
plt.show()
print(f"Best D_thr: {best_D_thr} with silhouette score: {best_score}")


from mpl_toolkits.mplot3d import Axes3D
# Συνάρτηση υπολογισμού απόστασης
def distance(point1, point2):
    return np.linalg.norm(point1 - point2)
# Συνάρτηση ανανέωσης κέντρου συστάδας
def update_cluster_center(cluster):
    return np.mean(cluster, axis=0)
# Συνάρτηση ανανέωσης ακτίνας συστάδας
def update_cluster_radius(cluster, center):
    return max([distance(point, center) for point in cluster])
# Συνάρτηση DBIECM
def dbiecm(data, D_thr):
    clusters = []
    cluster_centers = []
    cluster_radii = []
    for x_i in data:
        if not clusters:  # Δημιουργία της πρώτης συστάδας
            clusters.append([x_i])
```

```
            cluster_centers.append(x_i)
            cluster_radii.append(0)
        else:
            distances = [distance(x_i, center) for center in
cluster_centers]
            min_distance = min(distances)
            min_index = distances.index(min_distance)

            if min_distance > D_thr:
                # Δημιουργία νέας συστάδας
                clusters.append([x_i])
                cluster_centers.append(x_i)
                cluster_radii.append(0)
            else:
                # Προσθήκη στη υπάρχουσα συστάδα
                clusters[min_index].append(x_i)
                cluster_centers[min_index] =
update_cluster_center(clusters[min_index])
                cluster_radii[min_index] =
update_cluster_radius(clusters[min_index], cluster_centers[min_index])
    return clusters, cluster_centers, cluster_radii
# Μετατροπή σε NumPy array
data_array = df[['Column1._source.light', 'Column1._source.deep',
'Column1._source.rem']].values
# Ορισμός του ορίου
D_thr = 100
# Εφαρμογή του αλγορίθμου DBIECM
clusters, centers, radii = dbiecm(data_array, D_thr)
for i, (cluster, center, radius) in enumerate(zip(clusters, centers,
radii)):
    print(f"Cluster {i+1}:")
    print(f"  Points: {cluster}")
    print(f"  Center: {center}")
    print(f"  Radius: {radius}")
# Δημιουργία διαγραμμάτων
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
colors = ['r', 'g', 'b', 'y', 'c', 'm', 'k']
for i, cluster in enumerate(clusters):
    cluster = np.array(cluster)
    ax.scatter(cluster[:, 0], cluster[:, 1], cluster[:, 2], c=colors[i %
len(colors)], label=f'Cluster {i+1}')
    center = centers[i]
    radius = radii[i]
    ax.scatter(center[0], center[1], center[2], c='k', marker='x')
    # Plot the radius as a sphere
```

```
    u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
    x = center[0] + radius * np.cos(u) * np.sin(v)
    y = center[1] + radius * np.sin(u) * np.sin(v)
    z = center[2] + radius * np.cos(v)
    ax.plot_wireframe(x, y, z, color=colors[i % len(colors)], alpha=0.3)
ax.set_xlabel('Light')
ax.set_ylabel('Deep')
ax.set_zlabel('Rem')
ax.legend()
plt.show()

# Load and preprocess data
# Replace with your actual data file path
X = df[['Column1._source.light', 'Column1._source.deep',
'Column1._source.rem']].values
X = StandardScaler().fit_transform(X)  # Normalize data
# Define a range of D_thr values to test
D_thr_values = np.linspace(1, 100, 500)  # Example range; adjust based on
data scale
best_D_thr = None
best_score = -1
scores = []
valid_D_thr_values = [] # Store D_thr values for which silhouette score is
calculated
for D_thr in D_thr_values:
    # Run DBIECM
    clusters, cluster_centers, cluster_radii = dbiecm(X, D_thr)
    # Flatten clusters for silhouette score calculation
    labels = np.concatenate([np.full(len(cluster), i) for i, cluster in
enumerate(clusters)])
    # Compute silhouette score
    if len(clusters) > 1:  # Silhouette score requires at least 2 clusters
        score = silhouette_score(X, labels)
        scores.append(score)
        valid_D_thr_values.append(D_thr) # Store the corresponding D_thr
value
        if score > best_score:
            best_score = score
            best_D_thr = D_thr
# Plot silhouette scores using valid_D_thr_values
plt.plot(valid_D_thr_values, scores)
plt.xlabel('D_thr')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score vs D_thr')
plt.show()
print(f"Best D_thr: {best_D_thr} with silhouette score: {best_score}")
```

```
# Merge columns from different datasets
data1 = df[['Column1._source.deep', 'Column1._source.light',
'Column1._source.date']].values # Use 'data' instead of 'df'
df1 = pd.DataFrame(data1, columns=['deep', 'light', 'id']) # Assign
column names for clarity
# Sample DataFrame 2
data2 = data[['Column1._source.calories',
'Column1._source.date']].values
df2 = pd.DataFrame(data2, columns=['calories', 'id']) # Assign column
names for clarity
# Performing left join on 'id' column
merged_df = pd.merge(df1, df2, on='id', how='inner') # Use simplified
column names
print(merged_df)

# Συνάρτηση υπολογισμού απόστασης
def distance(point1, point2):
    return np.linalg.norm(point1 - point2)
# Συνάρτηση ανανέωσης κέντρου συστάδας
def update_cluster_center(cluster):
    return np.mean(cluster, axis=0)
# Συνάρτηση ανανέωσης ακτίνας συστάδας
def update_cluster_radius(cluster, center):
    return max([distance(point, center) for point in cluster])
# Συνάρτηση DBIECM
def dbiecm(data, D_thr):
    clusters = []
    cluster_centers = []
    cluster_radii = []
    for x_i in data:
        if not clusters:  # Δημιουργία της πρώτης συστάδας
            clusters.append([x_i])
            cluster_centers.append(x_i)
            cluster_radii.append(0)
        else:
            distances = [distance(x_i, center) for center in
cluster_centers]
            min_distance = min(distances)
            min_index = distances.index(min_distance)

            if min_distance > D_thr:
                # Δημιουργία νέας συστάδας
                clusters.append([x_i])
                cluster_centers.append(x_i)
                cluster_radii.append(0)
```

```
            else:
                # Προσθήκη στη υπάρχουσα συστάδα
                clusters[min_index].append(x_i)
                cluster_centers[min_index] =
update_cluster_center(clusters[min_index])
                cluster_radii[min_index] =
update_cluster_radius(clusters[min_index], cluster_centers[min_index])
    return clusters, cluster_centers, cluster_radii
# Μετατροπή σε NumPy array
data_array = merged_df[['calories', 'deep', 'light']].values
# Ορισμός του ορίου
D_thr = 300
# Εφαρμογή του αλγορίθμου DBIECM
clusters, centers, radii = dbiecm(data_array, D_thr)
for i, (cluster, center, radius) in enumerate(zip(clusters, centers,
radii)):
    print(f"Cluster {i+1}:")
    print(f"  Points: {cluster}")
    print(f"  Center: {center}")
    print(f"  Radius: {radius}")
# Δημιουργία διαγραμμάτων
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
colors = ['r', 'g', 'b', 'y', 'c', 'm', 'k']
for i, cluster in enumerate(clusters):
    cluster = np.array(cluster)
    ax.scatter(cluster[:, 0], cluster[:, 1], cluster[:, 2], c=colors[i %
len(colors)], label=f'Cluster {i+1}')
    center = centers[i]
    radius = radii[i]
    ax.scatter(center[0], center[1], center[2], c='k', marker='x')
    # Plot the radius as a sphere
    u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
    x = center[0] + radius * np.cos(u) * np.sin(v)
    y = center[1] + radius * np.sin(u) * np.sin(v)
    z = center[2] + radius * np.cos(v)
    ax.plot_wireframe(x, y, z, color=colors[i % len(colors)], alpha=0.3)
ax.set_xlabel('calories')
ax.set_ylabel('deep')
ax.set_zlabel('light')
ax.legend()
plt.show()
import numpy as np
import matplotlib.pyplot as plt

class DBIECM:
```

```
    def __init__(self, D_thr):
        self.D_thr = D_thr
        self.clusters = []
        self.centers = []
        self.radii = []
    def fit(self, systolic_data, diastolic_data):
        data = np.array(list(zip(systolic, diastolic)))

        # Step 1: Initialize the first cluster
        self.clusters.append([data[0]])
        self.centers.append(data[0])
        self.radii.append(0)
        for x_i in data[1:]:
            distances = [np.linalg.norm(x_i - center) for center in
self.centers]

            # Step 2: Calculate distances
            if all(distance > self.D_thr for distance in distances):
                # Step 4: Create new cluster
                self.clusters.append([x_i])
                self.centers.append(x_i)
                self.radii.append(0)
            else:
                # Step 3: Classify the data point
                min_index = np.argmin(distances)
                self.clusters[min_index].append(x_i)
                self.centers[min_index] =
np.mean(self.clusters[min_index], axis=0)
                self.radii[min_index] = max([np.linalg.norm(x -
self.centers[min_index]) for x in self.clusters[min_index]])
    def predict(self, systolic, diastolic):
        x_i = np.array([systolic, diastolic])
        distances = [np.linalg.norm(x_i - center) for center in
self.centers]
        min_distance = min(distances)
        if min_distance > self.D_thr:
            return -1  # Does not belong to any existing cluster
        else:
            return np.argmin(distances)
    def plot_clusters(self):
        colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k']
        plt.figure(figsize=(10, 6))
        for i, cluster in enumerate(self.clusters):
            cluster = np.array(cluster)
            plt.scatter(cluster[:, 0], cluster[:, 1], c=colors[i %
len(colors)], label=f'Cluster {i+1}')
```

```
            plt.scatter(self.centers[i][0], self.centers[i][1],
c=colors[i % len(colors)], marker='x', s=200)
        plt.xlabel('Systolic')
        plt.ylabel('Diastolic')
        plt.title('Clusters of Systolic and Diastolic Data')
        plt.legend()
        plt.show()
# Example data
np.random.seed(0)  # For reproducibility
systolic_data = np.random.normal(120, 15, 100)
diastolic_data = np.random.normal(80, 10, 100)
D_thr = 20
# Apply the algorithm
dbiecm = DBIECM(D_thr)
dbiecm.fit(systolic_data, diastolic_data)
# Plot the clusters
dbiecm.plot_clusters()

from collections import defaultdict
data = df[['Column1._source.light', 'Column1._source.deep',
'Column1._source.rem']].values
# Parameters
epsilon = 0.01  # estimation error
support_threshold = 0.02
w = int(1 / epsilon)
b = int(len(data) * support_threshold)
# TRIE
class TrieNode:
    def __init__(self):
        self.children = {}
        self.frequency = 0
        self.delta = 0
class Trie:
    def __init__(self):
        self.root = TrieNode()
    def insert(self, itemset):
        node = self.root
        # Convert NumPy array to a list of tuples for iteration
        for item in enumerate(itemset):
            if item not in node.children:
                node.children[item] = TrieNode()
            node = node.children[item]
        node.frequency += 1
    def update_set(self, b_current):
        self._update_set(self.root, b_current)
    def _update_set(self, node, b_current):
```

```
        if node.frequency + node.delta < b_current:
            return True  # Delete Node
        for child in list(node.children.keys()):
            if self._update_set(node.children[child], b_current):
                del node.children[child]
        return False
    def get_frequent_itemsets(self, min_frequency):
        frequent_itemsets = []
        self._get_frequent_itemsets(self.root, {}, frequent_itemsets,
min_frequency)
        return frequent_itemsets

    def _get_frequent_itemsets(self, node, current_set,
frequent_itemsets, min_frequency):
        if node.frequency >= min_frequency:
            frequent_itemsets.append((dict(current_set),
node.frequency))
        for item, child in node.children.items():
            current_set[item] = child
            self._get_frequent_itemsets(child, current_set,
frequent_itemsets, min_frequency)
            del current_set[item]
# Create Trie and Insert Data
trie = Trie()
b_current = 0
buffer = []
# Read and insert data in batches
for record in data:
    buffer.append(record)
    if len(buffer) >= w:
        # Transform current batch
        for entry in buffer:
            trie.insert(entry)
        # Update b_current
        b_current += 1
        # update data structure
        trie.update_set(b_current)
        # Empty buffer
        buffer = []
# Analyze results
frequent_itemsets = trie.get_frequent_itemsets(min_frequency=b)
print("Συχνά μοτίβα ύπνου:")
for itemset, frequency in frequent_itemsets:
    print(f"Μοτίβο: {itemset}, Συχνότητα: {frequency}")
```

```
from collections import defaultdict
data = df[['Column1._source.systolic.value',
'Column1._source.diastolic.value']].values
# parameters
epsilon = 0.0001  # estimation error
support_threshold = 0.0002
w = int(1 / epsilon)
b = int(len(data) * support_threshold)
# TRIE
class TrieNode:
    def __init__(self):
        self.children = {}
        self.frequency = 0
        self.delta = 0

class Trie:
    def __init__(self):
        self.root = TrieNode()

    def insert(self, itemset):
        node = self.root
        # Convert NumPy array to a list of tuples for iteration
        for item in enumerate(itemset):
            if item not in node.children:
                node.children[item] = TrieNode()
            node = node.children[item]
        node.frequency += 1

    def update_set(self, b_current):
        self._update_set(self.root, b_current)

    def _update_set(self, node, b_current):
        if node.frequency + node.delta < b_current:
            return True  # Delete node
        for child in list(node.children.keys()):
            if self._update_set(node.children[child], b_current):
                del node.children[child]
        return False
    def get_frequent_itemsets(self, min_frequency):
        frequent_itemsets = []
        self._get_frequent_itemsets(self.root, {}, frequent_itemsets,
min_frequency)
        return frequent_itemsets
    def _get_frequent_itemsets(self, node, current_set,
frequent_itemsets, min_frequency):
        if node.frequency >= min_frequency:
```

```
            frequent_itemsets.append((dict(current_set),
node.frequency))
        for item, child in node.children.items():
            current_set[item] = child
            self._get_frequent_itemsets(child, current_set,
frequent_itemsets, min_frequency)
            del current_set[item]
# create trie structure and insert data
trie = Trie()
b_current = 0
buffer = []
# Read and instert data in batches
for record in data:
    buffer.append(record)
    if len(buffer) >= w:
        # Transform current batch
        for entry in buffer:
            trie.insert(entry)
        # upload b_current
        b_current += 1
        # upload data structure
        trie.update_set(b_current)
        # Άδειασμα του buffer
        buffer = []
# Analyze results
frequent_itemsets = trie.get_frequent_itemsets(min_frequency=b)
print("Συχνά μοτίβα πίεσης:")
for itemset, frequency in frequent_itemsets:
    print(f"Μοτίβο: {itemset}, Συχνότητα: {frequency}")
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ελληνική

Χακίδη, Μ. Και Βαζιργιάννης, Μ. (2005). *Εξόρυξη γνώσης από βάσεις δεδομένων και τον παγκόσμιο ιστό*, Τυπωθήτω, Αθήνα.

Ξένη

Leskovec, A., Rajaraman, Ullman, J. (2014). *Mining of Massive Datasets*, Cambridge University Press, Cambridge

Mansour, M.,Abdullah, M. (2022). Mining Techniques for Streaming Data, *International Journal of Data Mining & Knowledge Management Process (IJDKP),* 2, 1-9.

Heymans, M. W., Twisk, W. R. J.(2022). Handling missibg data in clinical research, *Journal of Clinical Epidemiology*,4,185-188

Shukla, S. (2023). Real-time Monitoring and Predictive Analytics in Healthcare: Harnessing the Power of Data Streaming, *International Journal of Computer Applications*,8 ,33-36

Gama, J., Rodrigues, P. P. (2007). *Data Stream Processing, Learning from Data Streams - Processing Techniques in Sensor Networks,* Springer-Verlag, New York.

Pokrajac, D. , Lazarevic, A. and Latecki, L. J. (2007). Incremental Local Outlier Detection for Data Streams, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*,74,504-516

Coleman, B. (2022). Histograms with Exponential Decay on Streaming Data, *Randorithms*,4

Muhammad, H. (2024). Cluster Analysis – Types, Methods and Examples, *ResearchMethod.net*

Mingjing, D., Fuyu, W. (2022). Grid-Based Clustering Using Boundary Detection, *entropy,*11

Zhang, K., Zhong L., Tian, L., Zhang, X., Li, L. (2017). DBIECM-an Evolving Clustering Method for Streaming Data Clustering, *AMSE JOURNALS-AMSE IIETA*, 239-254

Ketan, R. S., Charles, N. (2020). Cluster Quality Analysis Using Silhouette Score, *IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*

Jiang, N., Grunewald, L. (2006). *Research Issues in Data Stream Association Rule Mining,* The University of Oklahoma, Norman

Manku, G. S.,Motwani, R. (2002). Approximate Frequency Counts over Data Streams, *Int'l Conf. on Very Large Databases*

Zhiguo, D., Minrui, F. (2013). An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window, *3rd IFAC International Conference on Intelligent Control and Automation Science*

Aitizaz, A., Hashim ,A., Aamir S., Aftab, A., Ting, T., Muhammad, A., Yazeed, Y., Heba, G. (2023). Blockchain-Powered Healthcare Systems: Enhancing Scalability and Security with Hybrid Deep Learning, *Sensors*

Zlatko, D. (2024). GDPR Considerations for Healthcare: Ensuring Data Protection Compliance, *GDPRLOCAL*