ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΠΡΟΗΓΜΕΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

«Μελέτη στη μηχανική μάθηση και την επιστήμη των δεδομένων με έμφαση στην υλοποίηση αλγορίθμων συσταδοποίησης με τη γλώσσα προγραμματισμού python»

Σπουδαστής: Αθανάσιος Μανωλιτζάς

Αριθμός Μητρώου: 22017

Μεταπτυχιακή Διπλωματική Εργασία

Αθήνα

Απρίλιος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΠΡΟΗΓΜΕΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

«Μελέτη στη μηχανική μάθηση και την επιστήμη των δεδομένων με έμφαση στην υλοποίηση αλγορίθμων συσταδοποίησης με τη γλώσσα προγραμματισμού python»

Σπουδαστής: Αθανάσιος Μανωλιτζάς

Αριθμός Μητρώου: 22017

Μεταπτυχιακή Διπλωματική Εργασία

Επιβλέπων καθηγητής: Μυλωνάς Φοίβος

Εγκρίθηκε από την τριμελή επιτροπή αξιολόγησης την ……

Φοίβος-Αποστολος Μυλωνάς - Ψηφιακά υπογεγραμμένο 2024.04.20 20:57:02 +03'00'

Kleio Sgouropoulou Digitally signed by Kleio Sgouropoulou Date: 2024.04.29 11:10:54 +03'00'

Christos Troussas Digitally signed by Christos Troussas Date: 2024.04.24 11:51:19 +03'00'

……	……	……
Α' μέλος	**Β' μέλος**	**Γ΄ μέλος**

Αθήνα

Απρίλιος 2024

UNIVERSITY OF WEST ATTICA

SCHOOL OF ENGINEERING

DEPARTMENT OF INFORMATION AND COMPUTER ENGINEERING

POSTGRADUATE PROGRAM MSC-ACS

ADVANCED COMPUTER SYSTEMS TECHNOLOGIES

Title of thesis:

« Study in machine learning and data science with an emphasis on implementing clustering algorithms with the python programming language »

Postgraduate student: Athanasios Manolitzas

Registration Number: 22017

Athens

April 2024

<u>ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ</u>

Ο κάτωθι υπογεγραμμένος Μανωλιτζάς Αθανάσιος του Αναστασίου, με αριθμό μητρώου mscacs22017, φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών «Προηγμένες Τεχνολογίες Υπολογιστικών Συστημάτων», του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, της Σχολής Μηχανικών, του Πανεπιστημίου Δυτικής Αττικής, βεβαιώνω ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου.»

Φοίβος-Απόστολος Μυλωνάς - Ψηφιακά υπογεγραμμένο
2024.04.20 20:56:38 +03'00'

Ο Δηλών

Περίληψη

Η επιστήμη για πολλά χρόνια, προσπαθεί να κατανοήσει και να εκλογικέψει τον τρόπο με τον οποίο ο άνθρωπος έχει την ικανότητα να σκέφτεται. Η προσπάθεια της κατανόησης της σκέψης του ανθρώπου δημιούργησε μια νέα επιστήμη που τα τελευταία χρόνια είναι ραγδαίως αναπτυσσόμενη, την επιστήμη της τεχνητής νοημοσύνης. Το πεδίο ανάπτυξης της επιστήμης αυτής είναι όχι μόνο η κατανόηση αλλά και η δημιουργία ευφυιών οντοτήτων. Υπάρχει σήμερα ένα μεγάλο εύρος υποπεδίων που εμπεριέχονται στην τεχνητή νοημοσύνη, με ένα από τα πιο βασικά αυτό της μηχανικής μάθησης.

Η μηχανική μάθηση καλύπτει την μελέτη αλγορίθμων υπολογιστικών συστημάτων των οποίων η απόδοση βελτιώνεται αυτόματα, μαθαίνοντας μέσω της εμπειρίας, με σκοπό την απόκτηση της ικανότητας της πρόβλεψης μέσω της μάθησης.

Τα δεδομένα που καλείται να διαχειριστεί το πεδίο της μηχανικής μάθησης δεν είναι φυσικά ούτε λίγα αλλά ούτε και απλά στη δόμησή τους. Το internet, οι συσκευές IoT και γενικά πολλά από αυτά που μας περιβάλλουν, δημιουργούν ένα τεράστιο όγκο πολύπλοκων δεδομένων, ο οποίος αυξάνεται εκθετικά με την πάροδο του χρόνου. Η μείωση της δυσκολίας και του κόστους συλλογής και διαχείρισης αυτών των δεδομένων γνωστών ως Big Data, προκειμένου να αξιοποιηθούν συντέλεσε στην ανάπτυξη του πεδίου της μηχανικής μάθησης.

Στο πρώτο μέρος της διπλωματικής αυτής, πραγματοποιείται μελέτη στην επιστήμη των δεδομένων και της μηχανικής μάθησης, αναλύοντας τις βασικές αρχές που τις διέπουν. Περιγράφονται οι βασικοί μέθοδοι και τεχνικές που συγκροτούν αυτές τις επιστήμες όπως είναι η μάθηση με και χωρίς επίβλεψη, καθώς και οι βασικοί αλγόριθμοι που χρησιμοποιούνται, όπως τα τεχνητά νευρωνικά δίκτυα, αλγόριθμοι κατηγοριοποίησης και συσταδοποίησης.

Στο δεύτερο μέρος δίνεται έμφαση στην υλοποίηση εφαρμογής στην γλώσσα προγραμματισμού python, η οποία χρησιμοποιεί την επιστήμη της μηχανικής μάθησης χωρίς επίβλεψη και ειδικότερα τον αλγόριθμο K-means με σκοπό την πρόβλεψη της οδηγικής συμπεριφοράς. Η δυνατότητα μοντελοποίησης της ρεαλιστικής οδηγικής συμπεριφοράς είναι χρήσιμη με εφαρμογές ποιότητας, που

είναι επωφελείς για τη βελτίωση της ασφάλειας του οδηγού, την αποφυγή τροχαίων ατυχημάτων, τη μείωση της κατανάλωσης καυσίμου και στις αυτοκινητοβιομηχανίες στο σύνολό τους.

Σκοπός της εφαρμογής είναι η αποκόμιση συμπερασμάτων, όπως αν η μηχανική μάθηση και ειδικότερα η μάθηση χωρίς επίβλεψη είναι ικανή να μοντελοποιήσει αξιόπιστα την ρεαλιστική οδηγική συμπεριφορά, χρησιμοποιώντας τα κατάλληλα εργαλεία, μεθόδους, τεχνικές και αλγόριθμους που προσφέρονται από την επιστήμη της μηχανικής μάθησης μέσω της γλώσσας προγραμματισμού της python, καταλήγοντας σε αυτά που είναι ιδανικότερα.

<u>Λέξεις κλειδιά</u>

τεχνητή νοημοσύνη, εξόρυξη δεδομένων, μηχανική μάθηση, μάθηση με επίβλεψη, μάθηση χωρίς επίβλεψη, τεχνητά νευρωνικά δίκτυα, πρόβλεψη οδηγικής συμπεριφοράς

Abstract

Science for many years has been trying to understand and rationalize the way humans have the ability to think. The effort to understand human thought has created a new science that has been developing rapidly in recent years, the science of artificial intelligence. The field of development of this science is not only understanding but also the creation of intelligent entities. There is currently a wide range of subfields involved in artificial intelligence, with one of the most fundamental being that of machine learning.

Machine learning covers the study of computational systems algorithms whose performance improves automatically, learning through experience, in order to acquire the ability to predict through learning.

The data that the field of machine learning is called upon to manage is of course neither few nor simple in its structure. The internet, IoT devices and generally much of what surrounds us, creates a huge volume of complex data, which grows exponentially over time. Reducing the difficulty and cost of collecting and managing this data, known as Big Data, in order to make use of it has contributed to the development of the field of machine learning.

At the first part of this diploma, a study is carried out in the science of data and machine learning, analyzing the basic principles that govern them. The basic methods and techniques that make up these sciences are described, such as learning with and without supervision, as well as the basic algorithms used, such as artificial neural networks, categorization and clustering algorithms.

The second part emphasizes the implementation of an application in the python programming language, which uses the science of unsupervised machine learning and in particular the K-means algorithm in order to predict driving behavior. The ability to model realistic driving behavior is useful with quality applications, which are beneficial for improving driver safety, avoiding traffic accidents, reducing fuel consumption, and the automotive industry as a whole.

The purpose of the application is to draw conclusions, such as whether machine learning and in particular unsupervised learning is able to reliably model realistic

driving behavior, using the appropriate tools, methods, techniques and algorithms offered by the science of machine learning through language of python programming, ending up with the ones that are most ideal.

Keywords

artificial intelligence, data mining, machine learning, supervised learning, unsupervised learning, artificial neural network, python, k-means, predict driving behavior

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΣΧΗΜΑΤΩΝ

ΠΑΡΑΡΤΗΜΑΤΑ ΚΩΔΙΚΑ

ΠΑΡΑΡΤΗΜΑΤΑ ΔΕΔΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΑΚΡΩΝΥΜΙΩΝ

ΑΚΡΩΝΥΜΙΟ	ΕΠΕΞΗΓΗΣΗ
ΤΝ	Τεχνητή Νοημοσύνη
ΤΝΔ	Τεχνητό Νευρωνικό Δίκτυο
GMM	Gaussian Mixture Models
FNN	Feedforward Neural Networks
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
ODR	Outliers Detection and Removal (Ανίχνευση και εξάλειψη ακραίων τιμών)
ND	Normalization Data (Κανονικοποίηση δεδομένων)
FS	Features Scaling (Κλιμάκωση χαρακτηριστικών)
PCA	Principal Component Analysis (Ανάλυση Κύριων Στοιχείων)
ULA	Unsupervised Learning Algorithm (Αλγόριθμος μάθησης χωρίς επίβλεψη)
EDA	Exploratory Data Analysis (Διερευνητική Ανάλυση Δεδομένων)

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Εισαγωγή

Για πολλά χρόνια, η επιστήμη προσπαθεί να κατανοήσει και να εκλογικέψει τον τρόπο με τον οποίο ο άνθρωπος έχει την ικανότητα να σκέφτεται. Την ικανότητα δηλαδή του ανθρώπου να αντιληφθεί, να ερμηνεύσει, να προβλέψει και να αλληλοεπιδράσει με όλα αυτά τα στοιχεία που αποτελούν έναν κόσμο πολύ μεγαλύτερο και πιο περίπλοκο από την ίδια την οντότητά της.
Η προσπάθεια της κατανόησης της σκέψης του ανθρώπου δημιούργησε μια νέα επιστήμη που τα τελευταία χρόνια είναι ραγδαίως αναπτυσσόμενη, την επιστήμη της τεχνητής νοημοσύνης (ΤΝ). Το πεδίο ανάπτυξης της επιστήμης αυτής είναι όχι μόνο η κατανόηση αλλά και η δημιουργία ευφυιών οντοτήτων.[1]

Τεχνητή Νοημοσύνη

Ένας ορισμός που χαρακτηρίζει την μελέτη των επιστημόνων πάνω στην ΤΝ και ο οποίος είναι διαχρονικός δίνεται από την Elaine Rich (1983):
«Η Τεχνητή Νοημοσύνη είναι η μελέτη του πώς να κάνετε τους υπολογιστές να κάνουν πράγματα στα οποία, αυτή τη στιγμή, οι άνθρωποι είναι καλύτεροι»
Υπάρχει σήμερα ένα μεγάλο εύρος υποπεδίων που εμπεριέχονται στην ΤΝ. Γενικά πεδία όπως η μάθηση και η αντίληψη και ειδικά πεδία όπως παιχνίδια σκέψης (π.χ. σκάκι), η απόδειξη μαθηματικών θεωρημάτων, η δημιουργία εικόνων, η δημιουργία μουσικής, η οδηγική συμπεριφορά και η οδήγηση αυτοκινήτου σε έναν πολυσύχναστο δρόμο, η διάγνωση ασθενειών κ.α.
Υπάρχουν διάφορες προσεγγίσεις της ΤΝ, που έχουν ακολουθηθεί όλα αυτά τα χρόνια, οι οποίες προσπαθούν να απαξιώσουν η μια την άλλη, αλλά ταυτόχρονα έχουν συμπληρώσει η μια την άλλη. Οι δύο γενικότερες προσεγγίσεις είναι η πιστότητα με μέτρο σύγκρισης τον άνθρωπο, ενώ η άλλη η πιστότητα με μέτρο σύγκρισης το ιδανικό που ονομάζεται ορθολογισμός. Η κάθε μια από αυτές τις

προσεγγίσεις καταπιάνεται με την διαδικασία της σκέψης και της συλλογιστικής, καθώς και με την διαδικασία της συμπεριφοράς.[1][2]

ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Εισαγωγή

Η εκτέλεση πολλαπλών σύνθετων υπολογισμών σε σύντομο χρονικό διάστημα είναι μια εργασία κατά την οποία τα υπολογιστικά συστήματα υπερτερούν έναντι των ανθρώπων. Από την άλλη η αναγνώριση ενός αγνώστου περιβάλλοντος, η λήψη άμεσων αποφάσεων και ο καθορισμός απαραίτητων ενεργειών, είναι εργασίες κατά τις οποίες ο ανθρώπινος εγκέφαλος υπερτερεί ακόμα έναντι των υπολογιστικών συστημάτων. Μια ιδιαίτερη δύναμη της ανθρώπινης νοημοσύνης, είναι η προσαρμοστικότητα. Ο ανθρώπινος εγκέφαλος είναι σε θέση να προσαρμοστεί σε διάφορες περιβαλλοντικές συνθήκες και βάση των συνθηκών να προσαρμόσει τη συμπεριφορά του μέσω της μάθησης. Η μαθησιακή ικανότητα του ανθρώπου είναι σαφώς ανώτερη από αυτή των υπολογιστικών συστημάτων και λαμβάνοντας υπόψη και τον ορισμό της Elaine Rich που αναφέραμε σε προηγούμενο κεφάλαιο, η μηχανική μάθηση καθίσταται ως ένα από τα βασικότερα υποπεδία της ΤΝ.[2]

Ο Alan Turing (1950) σχεδίασε ένα τεστ έτσι ώστε να δοθεί ορισμός για την νοημοσύνη. Στο τεστ Turing ένα υπολογιστικό σύστημα επιτυγχάνει, όταν σε κάποιες γραπτές ερωτήσεις που θέτονται από κάποιο άνθρωπο, δεν μπορεί να διακριθεί εάν οι γραπτές απαντήσεις που δίνονται προέρχονται από έναν άνθρωπο ή από έναν υπολογιστή.

Εκτός των άλλων δυνατοτήτων (επεξεργασία φυσικής γλώσσας, αναπαράσταση γνώσης, αυτοματοποιημένη συλλογιστική), για την επιτυχία του τεστ Turing, ένας υπολογιστής πρέπει να διαθέτει και την δυνατότητα της μηχανικής μάθησης, έτσι ώστε να μπορεί να προσαρμοστεί σε νέες συνθήκες και να ανιχνεύει μοτίβα. Οι δυνατότητες αυτές δημιουργούν την ικανότητα των υπολογιστικών συστημάτων να παρουσιάζουν και να εξηγούν με κατανοητούς όρους σε έναν άνθρωπο.[1]

Η μηχανική μάθηση καλύπτει την μελέτη αλγορίθμων υπολογιστικών συστημάτων των οποίων η απόδοση βελτιώνεται αυτόματα, μαθαίνοντας μέσω της εμπειρίας, με σκοπό την απόκτηση της ικανότητας της πρόβλεψης μέσω της μάθησης. Η ικανότητα της πρόβλεψης έγκειται στην ικανότητα να παράγει ακριβή συμπεράσματα σε νέα δεδομένα, αφού πρώτα έχει περάσει από την διαδικασία της εκπαίδευσης ενός

συνόλου γνωστών αντιπροσωπευτικών δεδομένων. Η διαδικασία της εκπαίδευσης παράγει ένα γενικό μοντέλο ανίχνευσης συγκεκριμένων μοτίβων (pattern recognition) που καθιστά δυνατή την παραγωγή προβλέψεων με επαρκή ακρίβεια. Για να είναι επιτυχημένη η εκπαίδευση θα πρέπει η πολυπλοκότητα των νέων δεδομένων να είναι αντίστοιχη της πολυπλοκότητας των γνωστών αντιπροσωπευτικών δεδομένων. Η μελέτη αυτή καθορίζει την μηχανική μάθηση σε λειτουργικό πλαίσιο παρά με γνωστικούς όρους. Επομένως η μηχανική μάθηση έρχεται και επιβεβαιώνει την πρόταση του Alan Turing αρκετά χρόνια μετά, αντικαθιστώντας το ερώτημα αν οι μηχανές μπορούν να αποκτήσουν την ικανότητα της σκέψης, με το ερώτημα αν οι μηχανές έχουν την ικανότητα να κάνουν αυτό που και ο άνθρωπος μπορεί να κάνει δηλαδή να σκεφτεί.[1] [4]

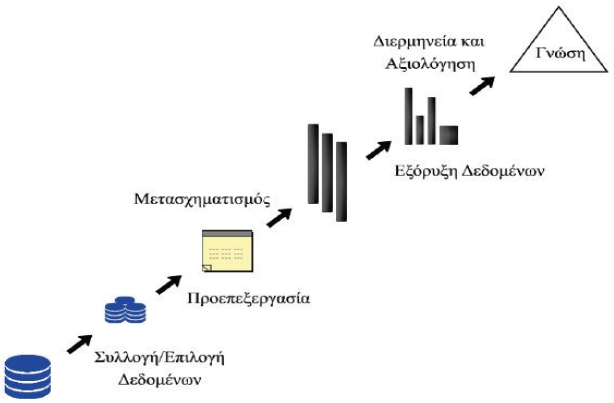
Τα δεδομένα που καλείται να διαχειριστεί το πεδίο της μηχανικής μάθησης δεν είναι ούτε λίγα αλλά ούτε και απλά στη δόμησή τους. Στις μέρες μας όπου το internet, οι συσκευές IoT και γενικά πολλά από αυτά που μας περιβάλλουν, δημιουργούν ένα τεράστιο όγκο πολύπλοκων δεδομένων, ο οποίος αυξάνεται εκθετικά με την πάροδο του χρόνου. Η μείωση της δυσκολίας και του κόστους συλλογής και διαχείρισης αυτών των δεδομένων γνωστών ως Big Data, προκειμένου να αξιοποιηθούν συντέλεσε στην ανάπτυξη του πεδίου της μηχανικής μάθησης.

Όπως προαναφέραμε το πεδίο της μηχανικής μάθησης βασίζεται στην ανίχνευση προτύπων (patterns). Τα πρότυπα αυτά αφορούν στον χαρακτηρισμό τους δηλαδή στον καθορισμό διάταξης και οργάνωσης της δομής των δεδομένων, είτε αυτή η διάταξη ήταν αναμενόμενη εκ των προτέρων είτε όχι. Η οργάνωση αυτή συντελείται με χρήση χαρακτηριστικών, γνωρισμάτων ή ιδιοτήτων που έχουν εξαχθεί από τα δεδομένα. Το αποτέλεσμα αυτής της διαδικασίας είναι η δημιουργία μοντέλων για την περιγραφή των δεδομένων, δηλαδή την εύρεση ομάδων αντικειμένων ή συστάδων με παρόμοια χαρακτηριστικά. Μια άλλη χρήση του αποτελέσματος είναι η πρόβλεψη, δηλαδή η χρήση τιμών γνωρισμάτων των δεδομένων, μέσω των οποίων μπορεί να εκτιμηθεί η μελλοντική ή άγνωστη τιμή ενός άλλου γνωρίσματος.[5]

Εξόρυξη Δεδομένων

Για να καταλήξουμε στην ανίχνευση προτύπων και να εξάγουμε το επιθυμητό αποτέλεσμα υπάρχει μια ολόκληρη διαδικασία που ονομάζεται ανακάλυψη γνώσης από βάσεις δεδομένων (Σχήμα 1), η οποία συντελείται μέσω της εξόρυξης δεδομένων και αποτελείται από τα εξής στάδια:

1. *Συλλογή Δεδομένων (Data Collection)*
   Η συλλογή και αποθήκευση των δεδομένων
2. *Προ-επεξεργασία Δεδομένων (Preprocessing)*
   Εύρεση και τακτοποίηση λαθών και ελλείψεων των δεδομένων, με χρήση συγκεκριμένων διαδικασιών.
3. *Μετασχηματισμός Δεδομένων (Transformation)*
   Κανονικοποίηση των δεδομένων σε κοινή μορφή η οποία περιλαμβάνει εξάλειψη τυχόν περιττών δεδομένων, σύμπτυξη δεδομένων, εξάλειψη θορύβου, δημιουργία νέων δεδομένων βασισμένων στα ήδη υπάρχοντα.
4. *Εξόρυξη Δεδομένων (Data Mining)*
   Χρησιμοποιούνται τα δεδομένα του προηγούμενου σταδίου για την δημιουργία μοντέλου με την χρήση ενός αλγορίθμου ικανό να κατηγοριοποιήσει ή να προβλέψει την τιμή ενός χαρακτηριστικού εφαρμόζοντάς το σε νέα-άγνωστα δεδομένα.
5. *Διερμηνεία και Αξιολόγηση (Interpretation / Evaluation)*
   Εκτελείται διερμηνεία και αξιολόγηση των δεδομένων που παράχθηκαν από την διαδικασία.

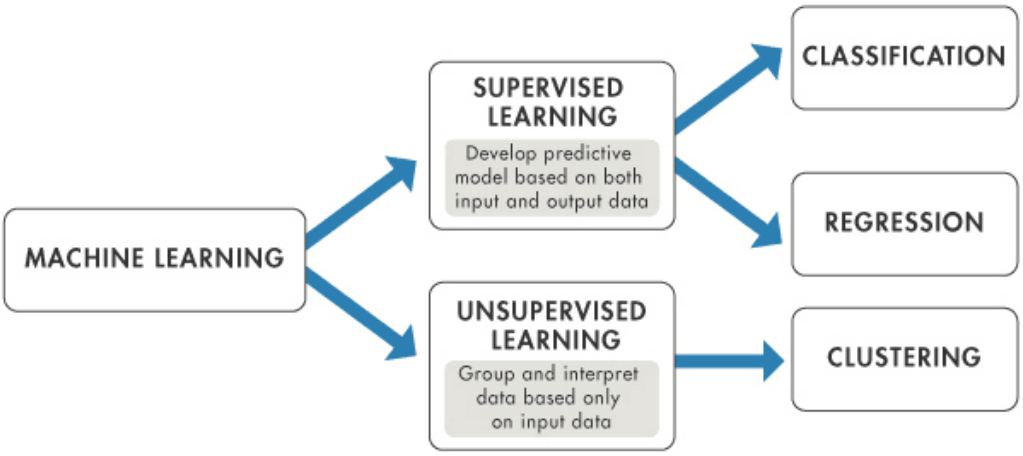


Σχήμα 1: Στάδια Ανακάλυψης Γνώσης από Βάσεις Δεδομένων

Η διάκριση των μοντέλων που δημιουργούνται από την διαδικασία της ανακάλυψης γνώσης από βάσεις δεδομένων γίνεται σε δύο τύπους, τα μοντέλα πρόβλεψης και τα περιγραφικά μοντέλα. Η διαφορά τους είναι ότι ενώ το μοντέλο πρόβλεψης κάνει εκτίμηση τιμών για ένα συγκεκριμένο χαρακτηριστικό και το οποίο βασίζεται στις τιμές άλλων χαρακτηριστικών, το περιγραφικό μοντέλο ανακαλύπτει πρότυπα των δεδομένων βάση των ιδιοτήτων τους, έτσι ώστε να αιτιολογηθεί η συμπεριφορά τους.[5]

Είδη Μηχανικής Μάθησης

Η μηχανική μάθηση είναι ένα ευρύ πεδίο που περιλαμβάνει πολλά διαφορετικά είδη αλγορίθμων και προσεγγίσεων. Ανάλογα με τον τρόπο που γίνεται η μάθηση και τον τρόπο με τον οποίο γίνεται η λήψη αποφάσεων, μπορούμε να κατηγοριοποιήσουμε τα είδη της μηχανικής μάθησης σε δύο βασικές κατηγορίες: μάθηση με επίβλεψη (Supervised Learning) και μάθηση χωρίς επίβλεψη (Unsupervised Learning).



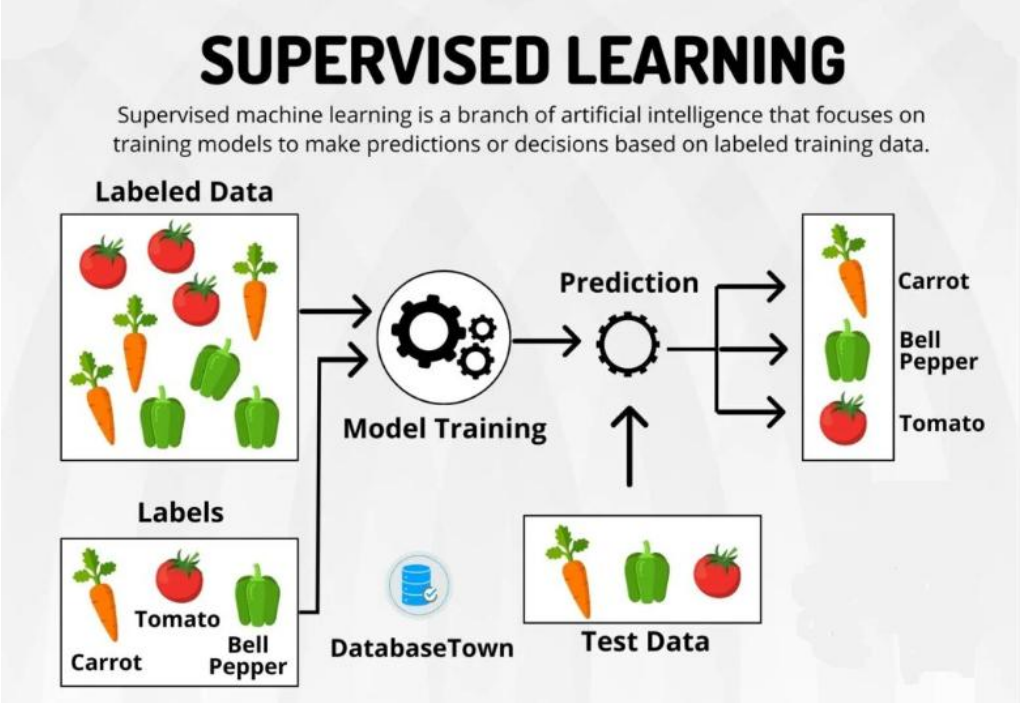
Σχήμα 2: Είδη Μηχανικής Μάθησης (φωτογραφία από MathWorks)

Κάθε κατηγορία μηχανικής μάθησης έχει τις δικές της εφαρμογές και προσεγγίσεις και μπορεί να χρησιμοποιηθεί ανάλογα με το πρόβλημα που προσπαθεί να λύσει ο ερευνητής ή ο μηχανικός.

Μάθηση με Επίβλεψη (Supervised Learning)

Στους αλγορίθμους που χρησιμοποιούνται στην μάθηση με επίβλεψη, ο υπολογιστής εκπαιδεύεται μέσω παραδειγμάτων. Για την εκπαίδευσή του αλγόριθμου χρειάζεται δηλαδή να τροφοδοτηθεί με δεδομένα για τα οποία υπάρχει η γνώση των δεδομένων εισόδου αλλά και των σωστών δεδομένων εξόδου αυτών των εισόδων. Ξέρουμε εκ των προτέρων τις ετικέτες (πόσες και ποιες είναι) της εξόδου. Αυτό που δεν είναι γνωστό είναι ο τρόπος με τον οποίο από την γνωστή είσοδο θα καταλήξουμε στην σωστή γνωστή έξοδο.

Για παράδειγμα, ας πούμε ότι διοχετεύουμε με δεδομένα εισόδου τον αλγόριθμο, τα οποία αποτελούνται με χαρακτηριστικά και γνωρίσματα λαχανικών όπως μήλων, καρότων και πιπεριών. Ο στόχος του αλγορίθμου είναι να εξάγει ως έξοδο, βασιζόμενος στα δεδομένα εισόδου, την σωστή απάντηση μήλο, καρότο ή πιπεριά. Φυσικά στην αρχή, που δεν έχει εκπαιδευτεί ο αλγόριθμος, το ποσοστό επιτυχίας θα είναι μικρό. Εάν η απάντηση που θα δώσει ο αλγόριθμος είναι λανθασμένη τότε συνεχίζει να εκπαιδεύεται και προσαρμόζει τις παραμέτρους του έτσι ώστε να βελτιώσει την ακρίβειά του και να ελαχιστοποιήσει τα σφάλματα του. Συγκεντρώνοντας όλο το σετ των δεδομένων εκπαίδευσης με την πάροδο του χρόνου θα αρχίσει να ανακαλύπτει και να καθορίζει πρότυπα βασιζόμενος σε ομοιότητες, διαφορές και άλλες λογικές μεταξύ των δεδομένων εισόδου, μέχρι να μπορεί να προβλέψει σωστά τις απαντήσεις στις ερωτήσεις μήλο, καρότο ή πιπεριά από μόνος του. Όσο πιο μεγάλος είναι ο όγκος των δεδομένων εκπαίδευσης που παρέχουμε στον αλγόριθμο που χρησιμοποιείται στην μάθηση με επίβλεψη, τόσο πιο εκπαιδευμένος θα καταστεί μέσω της εμπειρίας και θα μπορεί να προβλέψει σωστά αποτελέσματα, σε νέα άγνωστα δεδομένα, για τα οποία δεν γνωρίζουμε την απάντηση, με μεγαλύτερο ποσοστό ακρίβειας.



Σχήμα 3: Μάθηση με Επίβλεψη (φωτογραφία από Google Cloud)

Τα μοντέλα μάθησης με επίβλεψη χρησιμοποιούνται για την εκπαίδευση αλγορίθμων να επιτελούν ταξινόμηση δεδομένων ή προβλέψεις με ακρίβεια. Εφαρμόζεται σε πολλές περιπτώσεις όπως αναγνώριση εικόνων, αναγνώριση ομιλίας και συναισθημάτων, ανάλυση κυκλοφορίας, αξιολόγηση κινδύνου, ανίχνευση απάτης, φιλτράρισμα ανεπιθύμητων μηνυμάτων κ.α.[5] [6] [7] [8]

Τα πλεονεκτήματα των μοντέλων μάθησης με επίβλεψη είναι η πολύ καλή ακρίβεια στις προβλέψεις όταν η εκπαίδευση τους συντελείται σε ένα ποικίλο και αντιπροσωπευτικό σύνολο δεδομένων, η ευελιξία τους αφού μπορεί να εφαρμοστεί σε πληθώρα τομέων και τέλος στην ικανότητά τους να αποδίδουν ερμηνεύσιμα αποτελέσματα, κατανοητά από τους χρήστες.

Από την άλλη τα μειονεκτήματά τους έγκειται στην εξάρτηση που έχουν σε συγκεκριμένο σύνολο κατηγοριών, στην δυσκολία γενίκευσης σε νέα άγνωστα δεδομένα που μπορεί να διαφέρουν αρκετά από τα δεδομένα εκπαίδευσης και στην υπέρ-προσαρμογή που προκύπτει όταν η εκπαίδευση τους γίνεται σε περιορισμένο όγκο δεδομένων με αποτέλεσμα αντί να εκπαιδεύονται και να δημιουργούν μοτίβα, να απομνημονεύουν τα δεδομένα εκπαίδευσης, με αποτέλεσμα μιας καθόλου καλής απόδοσης κατά την εφαρμογή τους σε νέα άγνωστα δεδομένα.

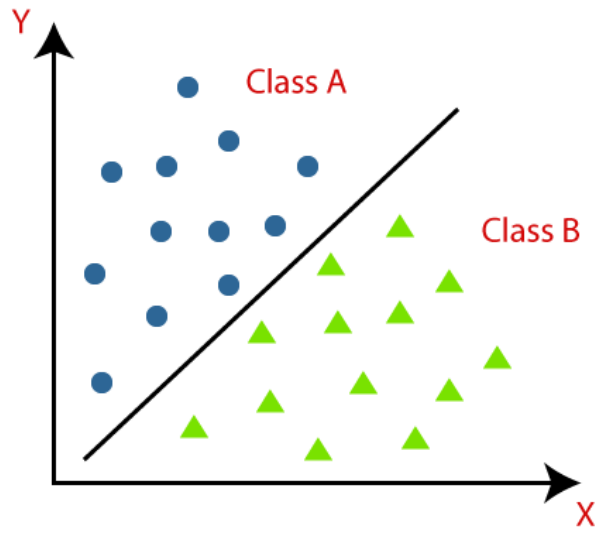
Για την αντιμετώπιση του προβλήματος της υπέρ-προσαρμογής μπορούμε είτε να μειώσουμε τον αριθμό των χαρακτηριστικών-γνωρισμάτων, είτε να κρατήσουμε όλα τα χαρακτηριστικά-γνωρίσματα, αλλά να μειώνουμε τα βάρη του αντίστοιχου χαρακτηριστικού κατά την εκπαίδευση και δημιουργία του μοντέλου. Η δεύτερη περίπτωση ονομάζεται κανονικοποίηση.

Η αξιολόγηση λοιπόν των μοντέλων μάθησης με επίβλεψη έγκειται στην ακρίβεια, ταχύτητα, ερμηνευσιμότητα, επεκτασιμότητα και ανθεκτικότητα ή την μέση τετραγωνική ρίζα σφάλματος.[5] [6]

Τα μοντέλα μάθησης με επίβλεψη διαχωρίζονται σε δύο βασικές κατηγορίες: την κατηγοριοποίηση (Classification) και την παρεμβολή (Regression).

Κατηγοριοποίηση (Classification)

Η κατηγοριοποίηση βασίζεται στην λογική ότι εκ των προτέρων υπάρχει η γνώση ενός προκαθορισμένου συνόλου από κατηγορίες και επίσης των δεδομένων που θα εισαχθούν ως είσοδος στον αλγόριθμο (χαρακτηριστικά και γνωρίσματα ενός αντικειμένου) και για τα οποία είναι γνωστό σε ποια κατηγορία ανήκουν.



Σχήμα 4: Μοντέλο Κατηγοριοποίησης (Classification)

Ο στόχος του αλγορίθμου της κατηγοριοποίησης είναι να εκπαιδευτεί επαρκώς έτσι ώστε όταν τροφοδοτηθεί με νέα άγνωστα δεδομένα για τα οποία δεν υπάρχει γνώση σε ποια κατηγορία ανήκουν, να μπορεί να εκτελέσει την κατηγοριοποίηση με σχετικά μεγάλη ακρίβεια.

Ένα παράδειγμα μοντέλου κατηγοριοποίησης, είναι η εκτίμηση πελατών τράπεζας που αιτούνται δάνειο για τον αν είναι ή όχι υψηλού κινδύνου. Η τράπεζα έχει στην κατοχή της στοιχεία των πελατών της που έλαβαν δάνειο από αυτήν, χαρακτηριστικά και γνωρίσματα όπως ηλικία, επάγγελμα, εισόδημα, καταθέσεις, υποθήκες κ.α. Επίσης γνωρίζει για το αν οι πελάτες αυτοί αποπλήρωσαν ή όχι το δάνειο που έλαβαν (κατηγορίες: χαμηλού και υψηλού κινδύνου). Έχει κατηγοριοποιήσει δηλαδή αναλόγως χαρακτηριστικών και γνωρισμάτων τους πελάτες της σε κατηγορίες χαμηλού και υψηλού κινδύνου. Βάσει αυτών των δεδομένων λοιπόν μπορεί να κάνει πρόβλεψη και να κατηγοριοποιήσει νέους, άγνωστους πελάτες με παρόμοια ή ίδια χαρακτηριστικά αν είναι χαμηλού ή υψηλού κινδύνου. Με αυτό τον τρόπο η τράπεζα υπολογίζει το ρίσκο αποδοχής της αίτησης δανείου από νέο υποψήφιο πελάτη και μπορεί να αποφασίζει αναλόγως. Επίσης γνωρίζοντας τα χαρακτηριστικά υποψηφίων με χαμηλό κίνδυνο, μπορεί να απευθυνθεί σε ομάδα ανθρώπων με ίδια χαρακτηριστικά μέσω της διαφήμισης.

Υπάρχουν διάφοροι αλγόριθμοι κατηγοριοποίησης. Οι βασικοί είναι οι εξής:

- *Δέντρα Απόφασης (Decision Trees)*

  Λαμβάνουν αποφάσεις κάνοντας διαχωρισμό των δεδομένων αναλόγως των χαρακτηριστικών – γνωρισμάτων τους, δημιουργώντας μια ιεραρχική δομή που μοιάζει με δέντρο και με αυτόν τον τρόπο κατηγοριοποιούν τα δεδομένα και κάνουν προβλέψεις. Γίνεται δηλαδή διαχωρισμός των δεδομένων σε όλο και μικρότερα υποσύνολα, αναπτύσσοντας σταδιακά με αυτό τον τρόπο ένα δέντρο αποφάσεων. Το τελικό αποτέλεσμα είναι ένα δέντρο με εσωτερικούς κόμβους, συνδέσεις μεταξύ των κόμβων και κόμβους φύλλων. Ο κάθε εσωτερικός κόμβος του δένδρου αντιπροσωπεύει ένα χαρακτηριστικό-γνώρισμα, η κάθε σύνδεση αντιπροσωπεύει μια συνθήκη ή τιμή για το χαρακτηριστικό-γνώρισμα του κόμβου από τον οποίο προέρχεται και κάθε φύλλο αντιπροσωπεύει μια κατηγορία.

- *Τυχαίο Δάσος (Random Forest)*

  Λαμβάνουν αποφάσεις συνδυάζοντας τις αποφάσεις από πολλαπλά δέντρα απόφασης, βελτιώνοντας με αυτό τον τρόπο την ακρίβεια της πρόβλεψης.

- *Κατηγοριοποιητής Bayes (Naive Bayes Classifier)*

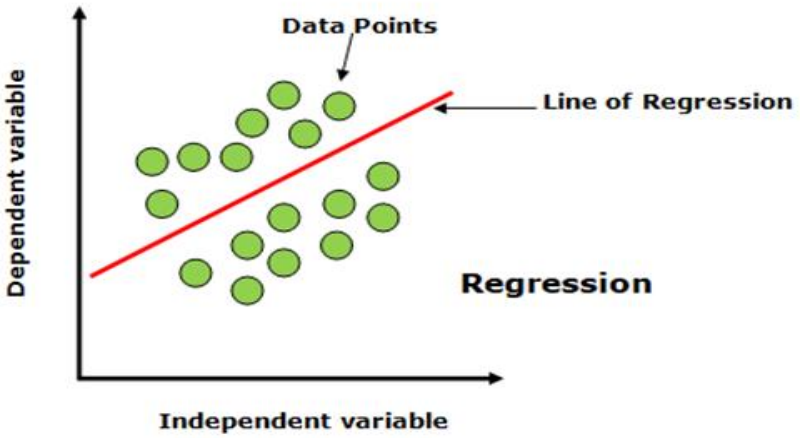
  Είναι βασισμένος στο θεώρημα του Bayes, χρησιμοποιώντας τις πιθανότητες για την κάθε κατηγορία να ανήκει στα χαρακτηριστικά – γνωρίσματα των

δεδομένων εισόδου. Δηλαδή υπολογίζει τις πιθανότητες των δεδομένων εισόδου να ανήκουν στην κάθε προδιαγεγραμμένη κατηγορία και τα κατατάσσει σε αυτήν με τις μεγαλύτερες πιθανότητες. Η συνήθη χρήση τους είναι σε εργασίες κατηγοριοποίησης.

- *Μηχανές διανυσμάτων υποστήριξης (Support Vector Machines)*
  Σκοπός του είναι η κατασκευή ενός υπέρ-επιπέδου, το οποίο διαχωρίζει τις κατηγορίες και βάση αυτού του διαχωρισμού θα λάβει αποφάσεις. Τα νέα δεδομένα κατηγοριοποιούνται ανάλογα με την πλευρά του υπέρ-επιπέδου στην οποία βρίσκονται. Ως καλύτερο υπέρ-επίπεδο θεωρείται αυτό που εξασφαλίζει το μέγιστο περιθώριο μεταξύ των κατηγοριών. Τα σημεία, τα οποία βρίσκονται στο όριο του περιθωρίου, ονομάζονται διανύσματα υποστήριξης. Τίθενται όρια, ικανά να μεγιστοποιούν τα περιθώρια των διαφορετικών κατηγοριών με αποτέλεσμα οι προβλέψεις να είναι ακριβείς.
- *Κοντινότερου Γείτονα (K-Nearest Neighbors)*
  Η απόφαση σε ποια κατηγορία θα ανήκουν τα νέα άγνωστα δεδομένα λαμβάνεται με βάση των k κοντινότερων γειτόνων των δεδομένων εκπαίδευσης.
- *Τεχνητά Νευρωνικά Δίκτυα (Artificial Neuron Network)*
  Είναι ένα δίκτυο τεχνητών νευρώνων διασυνδεδεμένων μεταξύ τους, οργανωμένους σε επίπεδα, που έχουν την ικανότητα μάθησης πολύπλοκων μοτίβων – προτύπων. Η λειτουργία του έγκειται στην αρχιτεκτονική του τεχνητού νευρωνικού δικτύου, ορίζοντας επίπεδα εισόδου, κρυφά επίπεδα και επίπεδα εξόδου. Τα επίπεδα εισόδου αντιστοιχούν στα χαρακτηριστικά-γνωρίσματα των δεδομένων εισόδου, τα κρυφά επίπεδα αποτελούνται από ένα ή περισσότερα επίπεδα νευρώνων, στους οποίους εφαρμόζονται μη γραμμικοί μετασχηματισμοί και το επίπεδο εξόδου το οποίο παρέχει την πρόβλεψη κατηγοριοποίησης. Το μοντέλο αυτό είναι ευέλικτο και μπορεί να χειριστεί πολύπλοκο και μεγάλο σε όγκο δεδομένα, ανιχνεύοντας πολύπλοκα μοτίβα.[5] [6] [7] [8]

Παρεμβολή (Regression)

Στην παρεμβολή υπάρχει η γνώση των δεδομένων που θα εισαχθούν ως είσοδος στον αλγόριθμο (χαρακτηριστικά και γνωρίσματα) αλλά ο στόχος της δεν είναι η κατανομή σε προκαθορισμένο σύνολο κατηγοριών. Ο στόχος της παρεμβολής είναι να πραγματοποιήσει πρόβλεψη μιας πραγματικής ή συνεχούς αριθμητικής τιμής, ανιχνεύοντας μια σχέση-μοτίβο μεταξύ δύο ή περισσότερων χαρακτηριστικών-γνωρισμάτων των δεδομένων εισόδου. Η αριθμητική τιμή που υπολογίζεται ονομάζεται εξαρτημένη μεταβλητή (Dependent Variable) , ενώ τα χαρακτηριστικά-γνωρίσματα των δεδομένων εισόδου που σχετίζονται με αυτήν, ονομάζονται ανεξάρτητες μεταβλητές (Independent Variables).



Σχήμα 5: Μοντέλο Παρεμβολής (Regression)

Ο τρόπος με τον οποίο επηρεάζεται η εξαρτημένη μεταβλητή από τις μεταβολές των ανεξάρτητων μεταβλητών, είναι κατανοητός στον χρήστη.

Για παράδειγμα, μπορεί να χρησιμοποιηθεί για την πρόβλεψη μισθού, λαμβάνοντας υπόψη ένα αντιπροσωπευτικό σετ δεδομένων με γνωρίσματα εργαζομένων, όπως έτη προϋπηρεσίας, αντικείμενο εργασίας, τοποθεσία και ισχύον μισθός. Όταν θα διεξαχθούν οι κατάλληλες ενέργειες εκπαίδευσης του αλγορίθμου θα μπορούσε το μοντέλο της παρεμβολής να πραγματοποιήσει πρόβλεψη του μέσου μισθού, βασισμένο στα γνωρίσματα που προαναφέρθηκαν.

Οι αλγόριθμοι παρεμβολής βάση των ικανοτήτων τους χρησιμοποιούνται σε μεγάλο βαθμό στην διεξαγωγή οικονομικών αναλύσεων.

Οι πιο βασικοί αλγόριθμοι παρεμβολής, είναι οι εξής:

- *Γραμμική Παρεμβολή (Linear Regression)*

  Στόχος του αλγορίθμου είναι η δημιουργία μιας γραμμικής σχέσης ανάμεσα στις ανεξάρτητες μεταβλητές και της εξαρτημένης μεταβλητής. Υπάρχουν παραλλαγές της γραμμικής παρεμβολής, όπως η πολλαπλή γραμμική παρεμβολή, στην οποία η εξαρτημένη μεταβλητή εξαρτάται από δύο ή περισσότερες ανεξάρτητες μεταβλητές και περιγράφεται ως ένα επίπεδο και η πολυωνυμική παρεμβολή η οποία περιγράφεται με ένα πολυώνυμο.
- *Παρεμβολή υποστήριξης διανυσμάτων (Support Vector Regression)*

  Είναι παραλλαγή του Support Vector Machines της κατηγοριοποίησης. Η διαφορά του έγκειται στο γεγονός ότι σκοπός του είναι η πρόβλεψη συνεχών αριθμητικών τιμών, αντί τον διαχωρισμό σε κατηγορίες. Ιδανικό για πρόβλεψη χρονοσειρών, τιμών μετοχών κ.α.
- *Δέντρα Απόφασης Παρεμβολής (Decision Trees Regression)*

  Είναι παραλλαγή των δέντρων απόφασης της κατηγοριοποίησης. Η διαφορά του έγκειται στο γεγονός ότι σκοπός του είναι η πρόβλεψη συνεχών αριθμητικών τιμών, αντί τον διαχωρισμό σε κατηγορίες. Στην ουσία ο αλγόριθμος διαχωρίζει το σύνολο δεδομένων σε δύο μέρη, βρίσκοντας το κατάλληλο σημείο της ανεξάρτητης μεταβλητής, τέτοιο ώστε το μέσο τετραγωνικό σφάλμα να είναι το ελαχιστοποιημένο. Επαναλαμβάνοντας αυτή την διαδικασία σχηματίζεται μια δομή που μοιάζει με δέντρο. Ο μέσος όρος των τιμών των φύλλων του δέντρου παράγει την εξαρτημένη μεταβλητή. Είναι επιρρεπής στην υπέρ-προσαρμογή. Χρησιμοποιείται σε δεδομένα όπου η σχέση μεταξύ ανεξαρτήτων μεταβλητών και εξαρτημένης δεν είναι γραμμική.
- *Παρεμβολή Λάσο (Lasso Regression)*

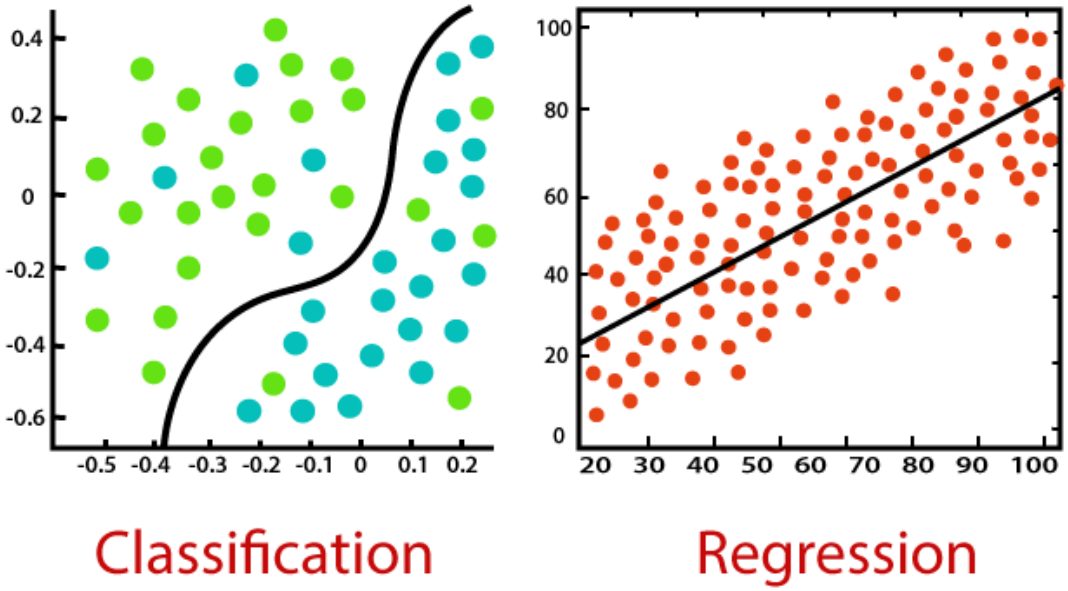
  Η τεχνική της παρεμβολής λάσο έχει την ικανότητα να πραγματοποιεί επιλογή γνωρισμάτων με δυνατότητα μηδενισμού ορισμένων συντελεστών, εκτελώντας με αυτόν τον τρόπο μια αυτόματη επιλογή χαρακτηριστικών. Αυτή η ικανότητα είναι χρήσιμη σε πολύπλοκα σύνολα δεδομένων με πολλά γνωρίσματα, καθώς και στον προσδιορισμό των καλύτερων δυνατόν ανεξαρτήτων μεταβλητών για δημιουργία σχέσεων με την εξαρτημένη μεταβλητή. Είναι ιδανική για την καταπολέμηση της υπερβολικής

προσαρμογής. Παράδειγμα εφαρμογής είναι όταν θέλουμε να κάνουμε πρόβλεψη πιθανότητας.

- *Παρεμβολή κορυφογραμμής (Ridge Regression)*
  Μοιάζει με την γραμμική παρεμβολή, με την διαφορά ότι σε αυτή την μέθοδο ελαχιστοποιείται η διακύμανση γραμμικής σχέσης μεταξύ των ανεξάρτητων και της εξαρτημένης μεταβλητής των δεδομένων. Πραγματοποιεί διόρθωση της υπέρ-προσαρμογής. Χρησιμεύει σε επίλυση προβλημάτων όπου τα διαθέσιμα δείγματα δεδομένων είναι λιγότερα από εκατό χιλιάδες ή όταν ο αριθμός των διαθέσιμων δειγμάτων είναι μικρότερος των χαρακτηριστικών-γνωρισμάτων.
- *Τεχνητά Νευρωνικά Δίκτυα (Artificial Neuron Network)*
  Παρόμοιο με τα τεχνητά νευρωνικά δίκτυα της κατηγοριοποίησης με την διαφορά ότι σκοπός του είναι η πρόβλεψη συνεχών αριθμητικών τιμών ως εξαρτημένη μεταβλητή σε σχέση με τις ανεξάρτητες μεταβλητές εισόδου, αντί τον διαχωρισμό σε κατηγορίες. Η λειτουργία του έγκειται στον ορισμό της αρχιτεκτονικής του τεχνητού νευρωνικού δικτύου ορίζοντας επίπεδα εισόδου, κρυφά επίπεδα και επίπεδο εξόδου. Τα επίπεδα εισόδου αντιστοιχούν στα χαρακτηριστικά-γνωρίσματα των δεδομένων εισόδου, τα κρυφά επίπεδα που αποτελούνται από ένα ή περισσότερα επίπεδα και στα οποία εφαρμόζονται μη γραμμικοί μετασχηματισμοί και το επίπεδο εξόδου το οποίο αποτελείται από έναν μόνο νευρώνα, ο οποίος παρέχει την πρόβλεψη παρεμβολής. Το μοντέλο αυτό είναι ευέλικτο και μπορεί να χειριστεί πολύπλοκα και μεγάλα σε όγκο δεδομένα, ανιχνεύοντας πολύπλοκα μοτίβα.[5] [6] [7] [8]

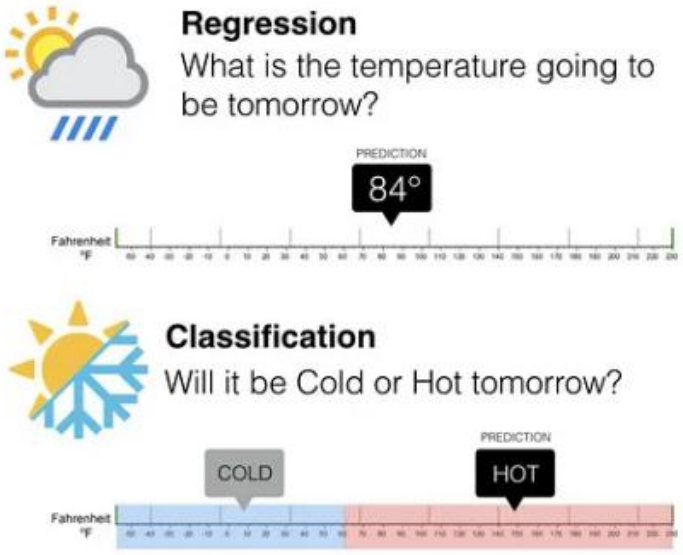
Διαφορές Κατηγοριοποίησης & Παρεμβολής

Εκ πρώτης όψεως δεν φαίνεται να υπάρχει διαφορά μεταξύ κατηγοριοποίησης (Classification) και παρεμβολής (Regression). Η βασική τους διαφορά είναι ότι ενώ η κατηγοριοποίηση χρησιμοποιείται για την πρόβλεψη διακριτών προκαθορισμένων κατηγοριών, η παρεμβολή χρησιμοποιείται για την πρόβλεψη μη διακριτής προκαθορισμένης συνεχούς αριθμητικής τιμής. Δηλαδή η κατηγοριοποίηση πραγματοποιεί πρόβλεψη μιας κατηγορίας, ενώ η παρεμβολή, πρόβλεψη μιας ποσότητας.



Σχήμα 6: Classification vs Regression

Για παράδειγμα η κατηγοριοποίηση μπορεί να χρησιμοποιηθεί για την πρόβλεψη αν θα έχει ζέστη ή κρύο αύριο, ενώ η παρεμβολή για την πρόβλεψη της θερμοκρασίας που θα έχει αύριο.

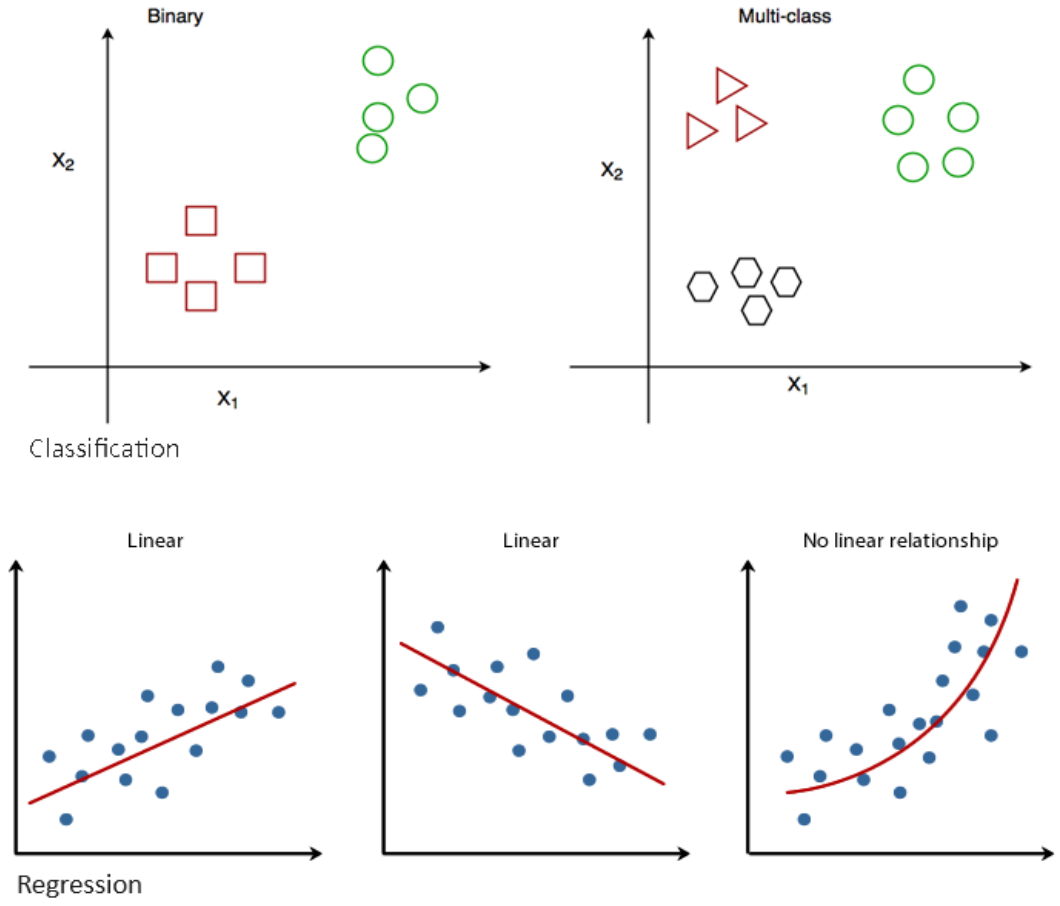


Σχήμα 7: Διαφορά πρόβλεψης Classification και Regression

Μια ακόμη διαφορά μεταξύ τους είναι στον τρόπο αξιολόγησής τους. Η παρεμβολή αξιολογείται βάση μέτρησης της μέσης τετραγωνικής ρίζας σφάλματος, ενώ η

κατηγοριοποίηση βάση μετρήσεων όπως ακρίβεια, ταχύτητα, ερμηνευσιμότητα, επεκτασιμότητα και ανθεκτικότητα.

Επίσης η παρεμβολή διαχωρίζεται σε γραμμική και μη γραμμική ενώ η κατηγοριοποίηση σε δυαδική (Binary Classifier) και πολυεπίπεδη (Multi-Class).[5] [6]



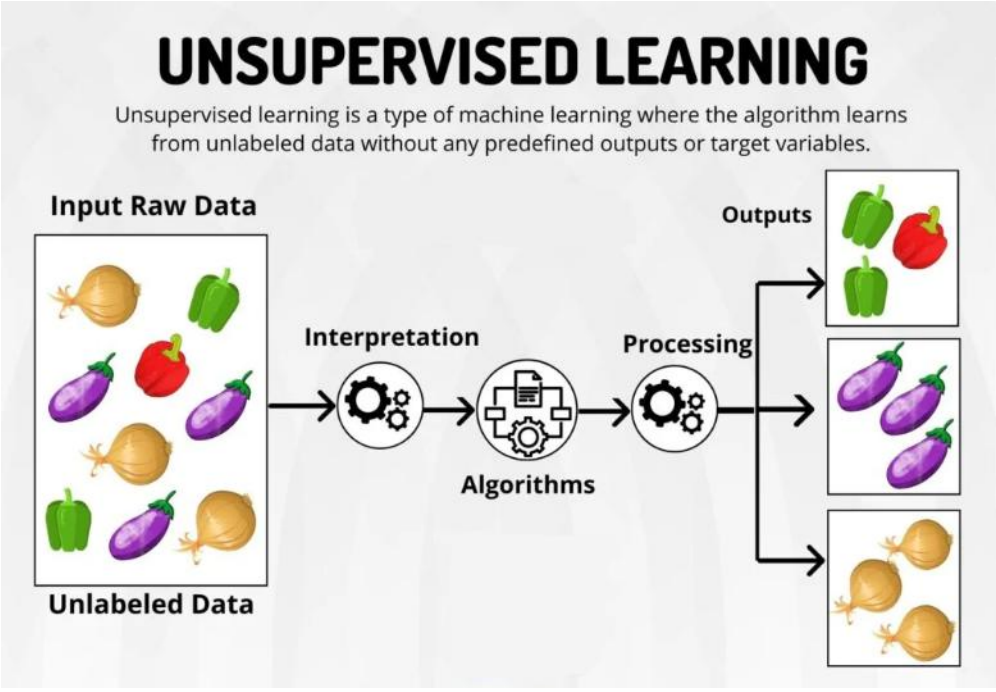
Σχήμα 8: Τύποι Classification & Regression

Μάθηση Χωρίς Επίβλεψη (Unsupervised Learning)

Οι αλγόριθμοι που χρησιμοποιούνται στη μάθηση χωρίς επίβλεψη, ανακαλύπτουν μοτίβα και πληροφορίες, χωρίς κάποια σαφή οδηγία και χωρίς να υπόκειται σε κάποια καθοδήγηση. Στα δεδομένα εισόδου δεν υπάρχει κατηγοριοποίηση εξ αρχής, δεν υπάρχει γνώση δηλαδή σε ένα πεπερασμένο σύνολο κατηγοριών, με αποτέλεσμα να μην υπάρχει γνώση των κατηγοριών εξόδου. Οι αλγόριθμοι της μάθησης χωρίς επίβλεψη, μαθαίνουν από μόνοι τους χωρίς να έχει προηγηθεί εκπαίδευση. Ο ρόλος τους είναι να ανακαλύπτουν τους δικούς τους κανόνες στα νέα-άγνωστα δεδομένα

που τους εισάγονται και να τα δομούν βασιζόμενοι σε διαφορές και ομοιότητες, χωρίς να τους παρέχονται οδηγίες και κανόνες του τρόπου που θα εργαστούν εκ των προτέρων. Δεν βασίζονται δηλαδή σε αντιστοιχίες είσοδος-προς-έξοδος, καθιστώντας την μάθηση χωρίς επίβλεψη, μη εξαρτημένη από επισημασμένα δεδομένα.

Για παράδειγμα, ας πούμε ότι διοχετεύουμε με δεδομένα εισόδου τον αλγόριθμο, τα οποία αποτελούνται από χαρακτηριστικά και γνωρίσματα λαχανικών που δεν έχουν εκ των προτέρων ετικέτες όπως φαίνεται στο παρακάτω σχήμα. Ο αλγόριθμος θα οργανώσει τα δεδομένα βασιζόμενος σε ομοιότητες και διαφορές που θα βρει. Το αποτέλεσμα θα είναι η δημιουργία 3 συστάδων αποτελεσμάτων χωρίς όμως να γνωρίζει τις ετικέτες τους, αλλά στην ουσία τα έχει διαχωρίσει σε είδος λαχανικών (πιπεριές, μελιτζάνες, κρεμμύδια). Θα μπορούσε να είχε ανακαλύψει ομοιότητες βάση χαρακτηριστικού χρώματος και να είχε δημιουργήσει 4 συστάδες αποτελεσμάτων, μια για την κόκκινη πιπεριά, μια για τις πράσινες πιπεριές, μια για τις μωβ μελιτζάνες και μια για τα καφέ κρεμμύδια.



Σχήμα 9: Μάθηση χωρίς Επίβλεψη (φωτογραφία από Google Cloud)

Οι ικανότητες των αλγορίθμων μάθησης χωρίς επίβλεψη, τους καθιστούν ιδανικούς για πολύπλοκες εργασίας όπως η οργάνωση τεραστίων σε όγκο δεδομένων σε συστάδες, όπως για παράδειγμα στην ομαδοποίηση εικόνων και κειμένου. Μια επίσης

μεγάλη ικανότητα που έχουν, είναι ο εντοπισμός μοτίβων σε δεδομένα, που δεν ήταν ορατά και ανιχνεύσιμα προηγουμένως, όπως για παράδειγμα, ο προσδιορισμός και προσέγγιση μέσω διαφήμισης, ομάδας πιθανών αγοραστών, στη διάθεση ενός καινούργιου προϊόντος. Ένα άλλο παράδειγμα, είναι η ανίχνευση ανωμαλιών στα δεδομένα, εντοπίζοντας μη φυσιολογικά μοτίβα ή παράταιρα στοιχεία, εντοπίζοντας με αυτό τον τρόπο απάτες, εισβολές σε συστήματα ή κατασκευαστικά ελαττώματα.

Το βασικό πλεονέκτημα λοιπόν των αλγορίθμων της μάθησης χωρίς επίβλεψη είναι η ελευθερία που έχουν να ανιχνεύουν μόνοι τους ομοιότητες, διαφορές και μοτίβα, χωρίς καθοδήγηση, ανακαλύπτοντας συνδέσεις και πληροφορίες που δεν ήταν αντιληπτές εκ των προτέρων. Για τον ίδιο λόγο προσφέρει επεκτασιμότητα, καθιστώντας αυτήν τη μέθοδο, ιδανική για τον χειρισμό τεραστίων σε όγκο δεδομένων, καθώς και όπως προαναφέραμε στην ανίχνευση ανωμαλιών. Τέλος σε κάποιες τεχνικές των αλγορίθμων της μάθησης χωρίς επίβλεψη, όπως η μείωση διαστάσεων (Dimensionality Reduction) που θα αναλύσουμε παρακάτω, πραγματοποιούν λειτουργίες προ-επεξεργασίας δεδομένων μειώνοντας θορύβους των δεδομένων και φιλτράροντας τα δεδομένα από άσχετα χαρακτηριστικά-γνωρίσματα, βελτιώνοντας με αυτό τον τρόπο την αποτελεσματικότητα.

Στον αντίποδα, ένα από τα μειονεκτήματα της μάθησης χωρίς επίβλεψη, είναι το γεγονός ότι μην έχοντας γνώση της εξόδου και προκαθορισμένου συνόλου κατηγοριών, δεν υπάρχει μέτρο αξιολόγησης της ακρίβειας των αποτελεσμάτων. Η αξιολόγηση καθίσταται υποκειμενική και έγκειται στην τεχνογνωσία των αρμοδίων. Για τον ίδιο λόγο, τα αποτελέσματα που παράγονται, δεν έχουν σαφή επεξηγήσεις στους χρήστες, στερώντας ερμηνευσιμότητα. Επίσης είναι επιρρεπή στην υπέρ-προσαρμογή. Τέλος λόγω της έλλειψης επίβλεψης, υπάρχει η πιθανότητα απόδοσης μοτίβων, μη επιθυμητών.

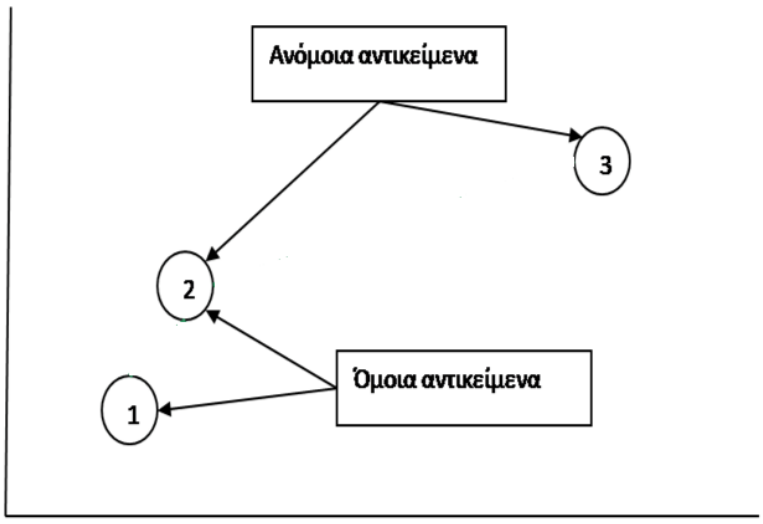
Οι βασικές τεχνικές των μοντέλων μάθησης χωρίς επίβλεψη είναι οι εξής:

- Συσταδοποίηση (Clustering)
- Κανόνες Συσχέτισης (Association Rules)

Συσταδοποίηση (Clustering)

Η συσταδοποίηση, είναι μια από τις πιο γνωστές τεχνικές της μάθησης χωρίς επίβλεψη, κατά την οποία, τα ακατέργαστα και χωρίς προκαθορισμένη ετικέτα δεδομένα, συγκροτούνται σε υποσύνολα δεδομένων, που ονομάζονται συστάδες (clusters), λαμβάνοντας ως βάση τις ομοιότητες τους. Τα δεδομένα της ίδιας συστάδας μοιάζουν μεταξύ τους και διαφέρουν από τα δεδομένα που ανήκουν στις άλλες συστάδες. Η τεχνική αυτή μοιάζει με την τεχνική της κατηγοριοποίησης της μάθησης με επίβλεψη με την διαφορά ότι στην κατηγοριοποίηση υπάρχει η γνώση του προκαθορισμένου συνόλου κατηγοριών-ομάδων στο οποίο ανήκουν τα δεδομένα, ενώ στη συσταδοποίηση, οι συστάδες που θα ενταχθούν τα δεδομένα, ανιχνεύονται από τον αλγόριθμο.

Ο τρόπος που λειτουργεί η τεχνική της συσταδοποίησης, είναι ο καθορισμός μέτρων ομοιότητας των δεδομένων. Ας υποθέσουμε ότι το σύνολο των δεδομένων έχει μόνο δύο χαρακτηριστικά-γνωρίσματα αριθμητικών τιμών. Κάθε δεδομένο από το σύνολο των δεδομένων, μπορεί να αναπαρασταθεί στο χώρο δύο διαστάσεων, ως ένα σημείο. Η απόσταση που χωρίζει το κάθε σημείο από το άλλο σε αυτό το χώρο, μπορεί να λειτουργήσει ως βαθμός ομοιότητας των δεδομένων. Αν ένα σημείο είναι κοντά σε κάποιο άλλο θα θεωρούνται όμοια και θα ανήκουν στην ίδια συστάδα, ενώ σε αντίθετη περίπτωση θα ανήκουν σε διαφορετική.



Σχήμα 10: Ομοιότητα βάση απόστασης

Εάν το σύνολο των δεδομένων έχει πάνω από δύο χαρακτηριστικά-γνωρίσματα τότε αυτά θα θεωρούνται σημεία στον χώρο τόσων διαστάσεων όσα και τα χαρακτηριστικά-γνωρίσματα. Υπάρχει διαφορετικός τρόπος υπολογισμού της απόστασης ο οποίος είναι ανάλογος του περιεχομένου των χαρακτηριστικών-γνωρισμάτων των δεδομένων (αριθμητικές, δυαδικές ή ονομαστικές τιμές). Για δεδομένα των οποίων τα χαρακτηριστικά-γνωρίσματα είναι αριθμητικά, το μέτρο της απόστασης μπορεί να είναι η Ευκλείδεια απόσταση μεταξύ δύο σημείων, σε έναν χώρο τόσων διαστάσεων όσων και των χαρακτηριστικών-γνωρισμάτων τους.

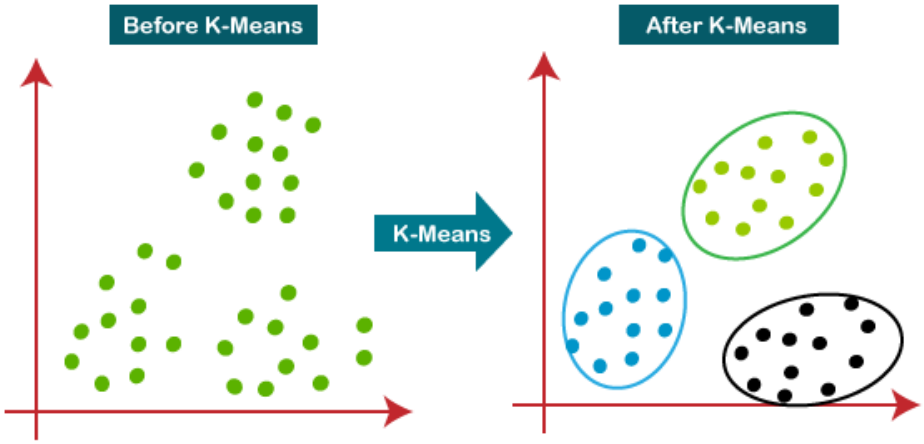
Η τεχνική της συσταδοποίησης, χρησιμοποιείται σε μια ποικιλία εφαρμογών, με μια από τις πιο γνωστές αυτή του μάρκετινγκ, επιμερίζοντας τους καταναλωτές με όμοια καταναλωτικά χαρακτηριστικά και με αυτό τον τρόπο να υπάρχει προς τον καθένα, μια πιο στοχευμένη διαφήμιση.

Οι αλγόριθμοι της συσταδοποίησης διαχωρίζονται στους εξής τύπους:

- *Διαχωριστικοί αλγόριθμοι (Partition-Based Clustering Algorithms)*
  Το κάθε σημείο από το σύνολο των δεδομένων ανήκει αποκλειστικά σε μία συστάδα. Υπάρχει ο διαχωρισμός βασισμένος σε κέντρο ή πρότυπο και ο διαχωρισμός βασισμένος στην πυκνότητα. Χαρακτηριστικό παράδειγμα αλγορίθμου διαχωριστικής συσταδοποίησης βασισμένος σε κέντρο, είναι ο αλγόριθμος K-μέσων (K-means).

  Αλγόριθμος K-means
  Βάση του αλγορίθμου K-means, το κάθε σημείο από το σύνολο των δεδομένων διαχωρίζεται σε μια από τις K σε αριθμό συστάδες, που έχουν οριστεί από τον χρήστη. Ο αλγόριθμος αποτελείται από μια επαναληπτική διαδικασία, κατά την οποία γίνεται υπολογισμός του κέντρου της κάθε συστάδας. Τα σημεία των δεδομένων που βρίσκονται πιο κοντά σε κάποιο

κέντρο, ανήκουν πλέον στην συστάδα του κέντρου αυτού.



Σχήμα 11: Αποκλειστικός Αλγόριθμος K-means

Στα πλεονεκτήματα αυτού του αλγόριθμου ανήκει η απλότητα, καθώς και η εύκολη κατανόησή του. Τα δεδομένα κατατάσσονται αυτόματα σε συστάδες και η υπολογιστική πολυπλοκότητα του αλγορίθμου είναι γραμμική, καθιστώντας τον αρκετά γρήγορο. Για τον λόγο αυτό, είναι ιδανικός για την διαχείριση μεγάλου όγκου δεδομένων.

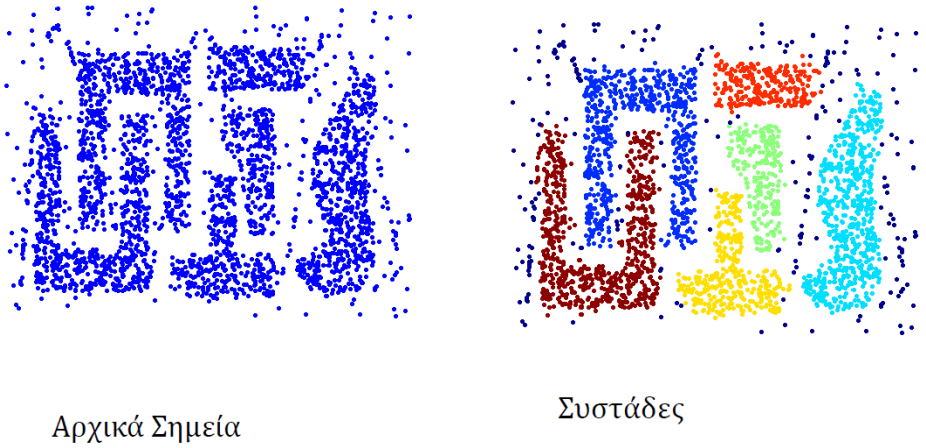
Από την άλλη, ένα από τα βασικά μειονεκτήματα του αλγόριθμου, είναι ότι ο αριθμός Κ των συστάδων, πρέπει να είναι προκαθορισμένος από τον χρήστη. Αν λοιπόν δεν γίνει σωστή αξιολόγηση του αριθμού των συστάδων που πρέπει να παραχθούν, το αποτέλεσμα μπορεί να μην είναι αξιόπιστο. Επίσης δεν λειτουργεί πολύ καλά, όταν στα δεδομένα υπάρχουν σημεία με ακραίες τιμές, διαφοροποιώντας με αυτό τον τρόπο αισθητά τα κέντρα των συστάδων, επηρεάζοντας την αξιόπιστη τελική παραγωγή τους. Τέλος, αφού από τον ορισμό κέντρων δημιουργείται η κάθε συστάδα, το αποτέλεσμα της μορφής της συστάδας, τείνει να είναι σφαιρική και ισομεγέθη, καθώς και δεν ανταποκρίνεται σωστά σε πολύπλοκα και με διαφορετικά μεγέθη δεδομένα.

Τα προβλήματα των αλγορίθμων βασισμένων σε κέντρα-πρότυπα, έρχονται να επιλύσουν μερικώς οι αλγόριθμοι βασισμένοι στη πυκνότητα. Η λογική της τεχνικής αυτής, βασίζεται στην ιδέα ότι μια συστάδα, αποτελείται από μια πυκνή περιοχή σημείων, η οποία διαχωρίζεται από τις άλλες συστάδες μέσω περιοχών αραιών σε πυκνότητα σημείων. Οι συστάδες που δημιουργούνται με

αυτόν τον τρόπο, είναι αυθαίρετου σχήματος και μεγέθους, χωρίς να είναι προκαθορισμένος ο αριθμός τους από τον χρήστη. Η μόνη απαραίτητη ενέργεια που απαιτείται, είναι ο καθορισμός μιας συνάρτησης, η οποία θα υπολογίζει την απόσταση μεταξύ των τιμών των δεδομένων, καθώς και το όριο απόστασης που θεωρείται κοντινή απόσταση.
Χαρακτηριστικό παράδειγμα αλγορίθμου διαχωριστικής συσταδοποίησης βασισμένος στη πυκνότητα, είναι ο αλγόριθμος DBSCAN (Density-Based Spatial Clustering of Applications with Noise).[5] [6] [8]

Αλγόριθμος DBSCAN

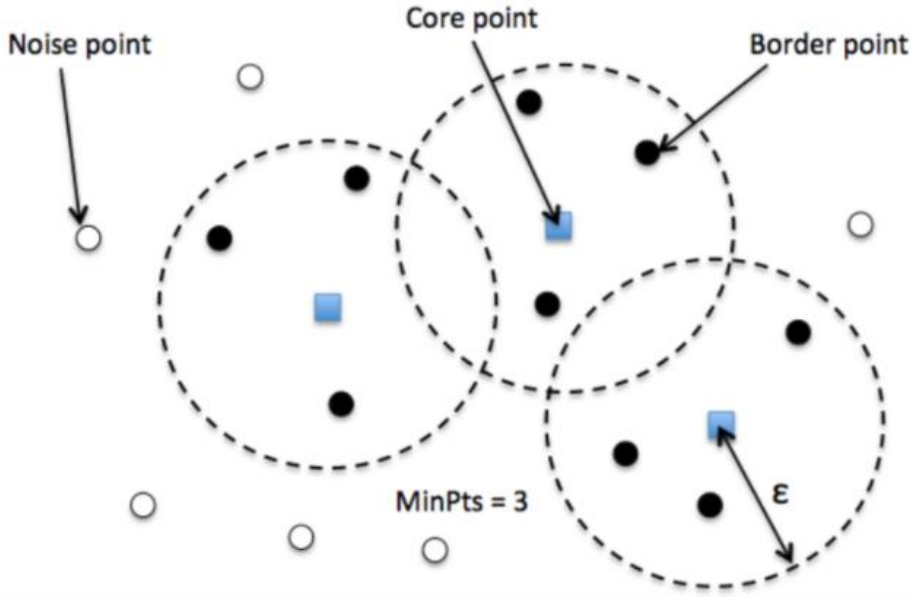
Ανιχνεύει συστάδες ανεξαρτήτου σχήματος και μεγέθους, από ένα μεγάλο σύνολο δεδομένων, το οποίο μπορεί να περιλαμβάνει ακραίες τιμές και θόρυβο.



Σχήμα 12: Διαχωριστικός αλγόριθμος βάση πυκνότητας (DBSCAN)

Γίνεται χρήση δύο παραμέτρων από τον αλγόριθμο. Η μια παράμετρος καθορίζει τον ελάχιστο αριθμό σημείων, που βρίσκονται σε κοντινή απόσταση μεταξύ τους, τέτοια ώστε να οριοθετούν μια πυκνή περιοχή (minPts). Η δεύτερη παράμετρος, καθορίζει το μέτρο ορίου απόστασης, το οποίο οριοθετεί την κοντινή απόσταση μεταξύ δύο σημείων (ε).
Ως πυκνή περιοχή που οριοθετεί μια συστάδα, ορίζεται το πλήθος των σημείων (minPts) μέσα σε μία καθορισμένη ακτίνα (ε). Υπάρχει ένα σημείο στη κάθε πυκνή περιοχή, που ονομάζεται πυρήνας (core point) και ο οποίος ορίζεται ως το σημείο, που περιέχει περισσότερα του minPts σημείων, μέσα

στην ακτίνα ε. Ένα σημείο περιθωρίου (border point), ορίζεται ένα σημείο, που έχει λιγότερα από MinPts μέσα σε ακτίνα ε, αλλά βρίσκεται στην γειτονιά ενός core point. Ένα σημείο θορύβου (noise point), ορίζεται οποιοδήποτε σημείο, που δεν είναι ούτε core point αλλά ούτε noise point.



Σχήμα 13: Παράμετροι DBSCAN αλγόριθμου

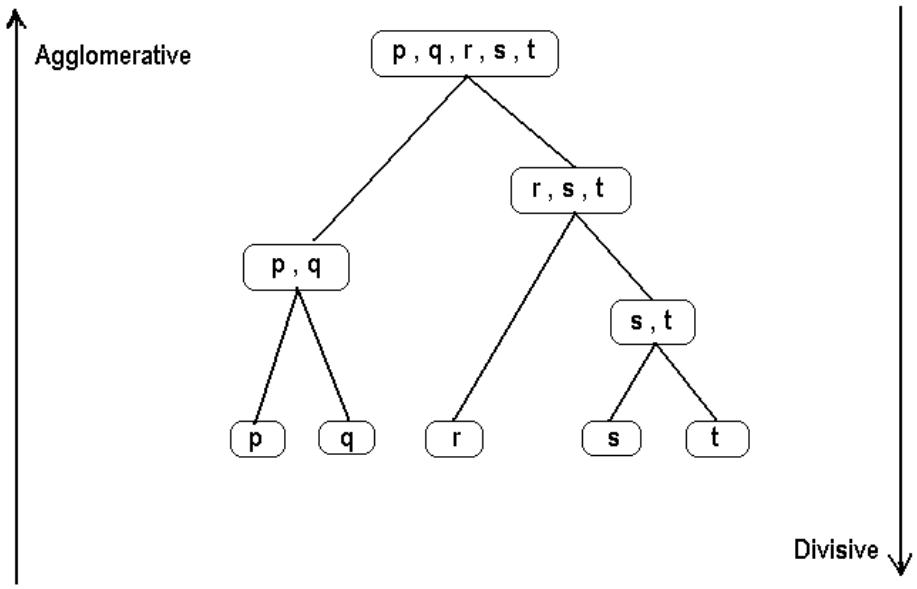
Στα πλεονεκτήματα του αλγορίθμου, είναι ότι δεν υπάρχει προκαθορισμένος αριθμός συστάδων, μπορεί να προσδιορίσει συστάδες με αυθαίρετο σχήμα και μέγεθος, ακόμα και μια συστάδα γύρω από μια άλλη συστάδα, με αποτέλεσμα να είναι ιδανικός για δεδομένα με ακραίες τιμές και θόρυβο και χρειάζεται ο προσδιορισμός μόνο δύο παραμέτρων από τον χρήστη, ο οποίος εάν έχει μελετήσει και κατανοήσει τα δεδομένα, δεν καθίσταται δύσκολος.

Στον αντίποδα, μειονεκτήματα του αλγόριθμου, είναι η κακή απόδοσή του σε δεδομένα με μεγάλες διαφορές πυκνότητας, καθώς και η ασάφεια που μπορεί να προκύψει, για την κατοχή περιθωριακών σημείων (noise points) από μια συστάδα ή γειτονική της.[5]

- *Ιεραρχικοί αλγόριθμοι (Hierarchical Clustering Algorithms)*

  Βασίζονται σε μια διαδικασία, κατά την οποία γίνεται συγχώνευση ή διάσπαση συστάδων. Υπάρχουν δύο κύριοι τύποι ιεραρχικής συσταδοποίησης: η αθροιστική (Agglomerative Clustering) και η διαιρετική συσταδοποίηση

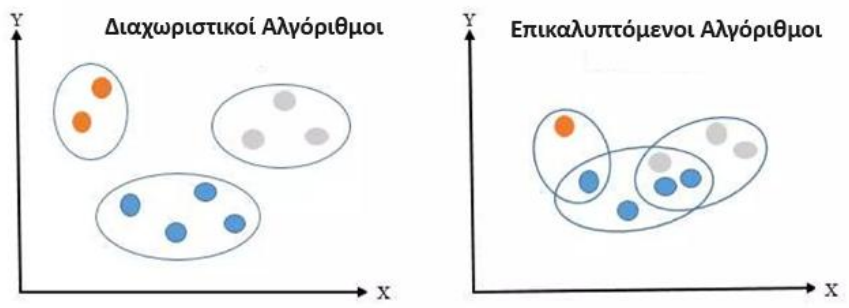
(Divisive Clustering). Στην αθροιστική συσταδοποίηση, δημιουργείται μια ιεραρχική δομή συστάδων, από κάτω προς τα πάνω, ενώ στην διαιρετική δημιουργείται ιεραρχική δομή συστάδων, από πάνω προς τα κάτω. Ποιο επίπεδο της ιεραρχικής δομής, είναι το κατάλληλο αποτέλεσμα, κρίνεται από τον χρήστη.



Σχήμα 14: Ιεραρχική Συσταδοποίηση (Αθροιστική - Διαιρετική)

Η αθροιστική συσταδοποίηση, βασίζεται στην συγχώνευση των μεταξύ δύο πλησιέστερων συστάδων. Κατά την έναρξη του αλγόριθμου, κάθε σημείο του συνόλου δεδομένων ορίζεται ως μια ξεχωριστή συστάδα. Εκτελούνται επαναλήψεις στον αλγόριθμο, πραγματοποιώντας ένωση των δύο πλησιέστερων συστάδων (αυτών με την μεγαλύτερη ομοιότητα). Ο αλγόριθμος τερματίζει, όταν όλα τα σημεία ανήκουν σε μια συστάδα. Η διαιρετική συσταδοποίηση λειτουργεί ακριβώς αντίθετα. Κατά τη έναρξη του αλγορίθμου, όλα τα σημεία του συνόλου δεδομένων ανήκουν σε μια συστάδα. Στη συνέχεια η μία και μοναδική συστάδα, διασπάται σε δύο διαφορετικές συστάδες. Η διάσπαση εκτελείται με κριτήριο, ότι οι συστάδες που θα δημιουργηθούν, θα είναι όσο το δυνατόν ανόμοιες μεταξύ τους. Η διαδικασία επαναλαμβάνεται, έως ότου το κάθε σημείο του συνόλου των δεδομένων, να είναι και μια ξεχωριστή συστάδα.[5] [6] [7] [8]

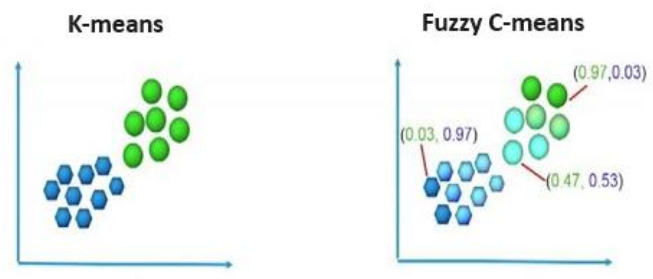
- *Επικαλυπτόμενοι αλγόριθμοι (Overlapping Clustering Algorithms)*
  Αποτελεί μια χαλαρή τεχνική συσταδοποίησης, κατά την οποία, κάθε ένα σημείο από τα δεδομένα, μπορεί να ανήκει σε πολλές συστάδες ταυτοχρόνως, με διαφορετικούς βαθμούς συμμετοχής.



Σχήμα 15: Διαφορά Διαχωριστικών και Επικαλυπτόμενων Αλγορίθμων

Το κάθε σημείο των δεδομένων λοιπόν, έχει ένα βαθμό συμμετοχής στη κάθε συστάδα. Αυτό μπορεί να έχει ως αποτέλεσμα την συγκρότηση επικαλυπτόμενων συστάδων. Το άθροισμα αυτών των βαθμών πρέπει να ισούται με την μονάδα, οπότε κάθε βαθμός, στην ουσία υποδηλώνει και μια πιθανότητα, το σημείο να ανήκει σε κάποια συστάδα.
Χαρακτηριστικός αλγόριθμος αυτής της τεχνικής, είναι ο αλγόριθμος fuzzy c-means.

Αλγόριθμος fuzzy c-means
Ο αλγόριθμος αυτός, αποτελεί παραλλαγή του K-means. Στην παραλλαγή αυτή, το κέντρο ορίζεται, ως ο υπολογισμός του μέσου όρου όλων των παραδειγμάτων, σταθμισμένος με το βαθμό συμμετοχής, που ανήκουν στη συστάδα.



Σχήμα 16: Διαφορά K-means με Fuzzy C-means

Ένα από τα πλεονεκτήματα αυτού του αλγόριθμου, είναι ότι επιτρέπει σε ένα σημείο να ανήκει σε πολλαπλές συστάδες, αναπαριστώντας πιο φυσικά την συμπεριφορά του πραγματικού κόσμου, όπου τα δεδομένα έχουν εμπλοκή σε πολλαπλές και όχι μία περιπτώσεις.

Στα μειονεκτήματα του αλγόριθμου, συγκαταλέγονται ότι είναι αναγκαίο να προκαθοριστεί το c, ο αριθμός των συστάδων και ότι δεν έχει ντετερμινιστικά αποτελέσματα.[5] [6]

- *Πιθανοκρατικοί αλγόριθμοι (Probabilistic Clustering Algorithms)*

  Οι αλγόριθμοι αυτοί, βασίζονται σε μοντέλα πιθανοτήτων για την συγκρότηση των συστάδων, αναπαριστώντας τις πιθανότητες των δεδομένων, να ανήκουν σε κάθε συστάδα. Με αυτή την τεχνική, ένα δείγμα δεδομένων, μπορεί να έχει μία υψηλή πιθανότητα να ανήκει σε μία συστάδα, αλλά ταυτόχρονα, μια μικρή πιθανότητα να ανήκει και σε άλλες συστάδες.

  Ένα παράδειγμα πιθανοκρατικού αλγορίθμου, είναι ο αλγόριθμος μείξης μοντέλων Gauss (*Gaussian Mixture Models - GMM*). Σε ένα GMM, κάθε συστάδα περιγράφεται από μία ή περισσότερες συναρτήσεις Gauss κατανομής πιθανότητας. Τα δεδομένα, εκφράζονται ως συνδυασμός αυτών των συναρτήσεων, με κάθε συνάρτηση να έχει ένα βάρος, που αντιστοιχεί στην πιθανότητα ενός δείγματος, να ανήκει σε αυτήν τη συστάδα.

  Οι πιθανοκρατικοί αλγόριθμοι, είναι χρήσιμοι σε περιπτώσεις όπου υπάρχει αβεβαιότητα, σχετικά με την ανάθεση δεδομένων σε μια συγκεκριμένη συστάδα, δηλαδή όταν έχουμε επικαλυπτόμενες συστάδες.

  Εκ πρώτης όψης, μπορεί ο αλγόριθμος GMM να συγχέεται με τον αλγόριθμο fuzzy c-means, γιατί χρησιμοποιούν και οι δύο τις πιθανότητες. Η διαφορά, έγκειται στο γεγονός, ότι ο αλγόριθμος GMM υποθέτει, ότι τα δεδομένα προέρχονται από μία μίξη κανονικών κατανομών (Gaussians), αντιπροσωπεύοντας κάθε συστάδα, από ένα κέντρο και μία κανονική κατανομή, ενώ ο αλγόριθμος fuzzy c-means, χρησιμοποιεί μία προσέγγιση με κέντρα συστάδας και "βαρύτητες", που αντιπροσωπεύουν την πιθανότητα ένα δεδομένο να ανήκει σε κάθε συστάδα. Ο GMM επικεντρώνεται στην πιθανοφάνεια των δεδομένων και ο fuzzy c-means αντιμετωπίζει την ασάφεια στην ομαδοποίηση. [5] [6] [7] [8]

Κανόνες Συσχέτισης (Association Rules)

Οι αλγόριθμοι των κανόνων συσχέτισης, λειτουργούν για τον εντοπισμό συσχετίσεων μεταξύ διαφορετικών στοιχείων, σε μεγάλα σύνολα δεδομένων, δηλαδή στην εύρεση μοτίβων από στοιχεία, που εξαρτώνται το ένα από το άλλο και τη χαρτογράφηση των συνδέσεων μεταξύ τους. Μπορεί να ανιχνεύσει ενδιαφέρουσες σχέσεις, σχετικά με το πώς και γιατί συνδέονται δύο στοιχεία.

Η λογική των κανόνων συσχέτισης, είναι η ανίχνευση κανόνων της μορφής "Αν Χ τότε Υ" (Χ→Υ), όπου Χ και Υ είναι σύνολα αντικειμένων ή γνωρισμάτων. Για παράδειγμα, ένας κανόνας συσχέτισης του τύπου {καφέ}→{γλυκό}, υποδηλώνει ότι αν κάποιος αγοράσει καφέ, τότε πιθανόν να αγοράσει και γλυκό.

Η μέτρηση της ποιότητας των κανόνων συσχέτισης, πραγματοποιείται με δύο βασικά ποσοτικά μεγέθη, την υποστήριξη (support) και την εμπιστοσύνη (confidence).

Η υποστήριξη, είναι η πιθανότητα που ένα στοιχείο εμφανίζεται στο σύνολο δεδομένων. Υπολογίζεται ως ο λόγος του αριθμού των εγγραφών των δεδομένων που περιέχουν το στοιχείο, προς το σύνολο των δεδομένων.

$$Support(X) = \frac{count(\text{εγγραφές δεδομένων που περιέχουν } X)}{sum(\text{εγγραφών δεδομένων})}$$

Η εμπιστοσύνη, αντιπροσωπεύει την πιθανότητα που η συσχέτιση μεταξύ των στοιχείων του Χ και του Υ (Χ→Υ), είναι ακριβής. Υπολογίζεται ως ο λόγος του αριθμού των εγγραφών, που περιέχουν τόσο το Χ όσο και το Υ στοιχείο, προς τον αριθμό των εγγραφών που περιέχουν το Χ στοιχείο.

$$Confidence(X \rightarrow Y) = \frac{count(\text{εγγραφών που περιέχουν } X \text{ και } Y)}{sum(\text{εγγραφών που περιέχουν } X)}$$

Σημαντική ιδιότητα για τους κανόνες συσχέτισης, είναι η ιδιότητα “a priori”, κατά την οποία αν ένα σύνολο στοιχείων θεωρείται συχνό, τότε όλα τα υποσύνολά του θεωρούνται συχνά. Όταν λέμε συχνό, εννοούμε το ποσοστό των εγγραφών των δεδομένων, που περιέχονται τα ζητούμενα στοιχεία και το οποίο είναι ίσο ή μεγαλύτερο ενός ορίου, που ορίζεται από τον χρήστη.

$$Rule: X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

Example:

Rule	Support	Confidence	Lift
$A \Rightarrow D$	2/5	2/3	10/9
$C \Rightarrow A$	2/5	2/4	5/6
$A \Rightarrow C$	2/5	2/3	5/6
$B \& C \Rightarrow D$	1/5	1/3	5/9

Σχήμα 17: Κανόνες Συσχέτισης (Association Rules)

Οι κανόνες συσχέτισης είναι χρήσιμοι σε πολλές εφαρμογές, όπως για την ανίχνευση προτιμήσεων των καταναλωτών και την πρόβλεψη συμπεριφοράς αγοραστών.
Από την ιδιότητα "a priori" που προαναφέραμε, υπάρχει ο ομώνυμος και πιο γνωστός αλγόριθμος των κανόνων συσχέτισης, ο αλγόριθμος «A Priori».

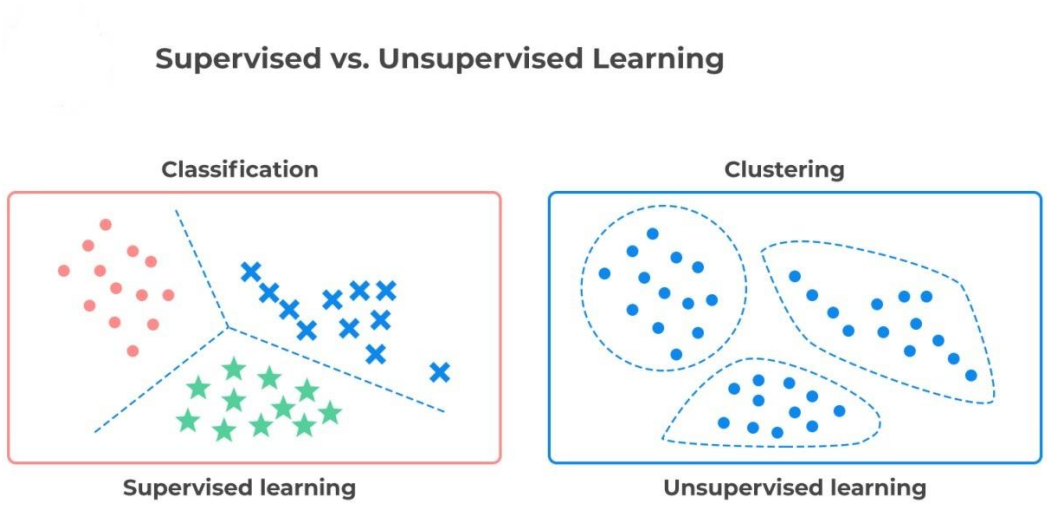
Αλγόριθμος «A Priori»

Ο βασικός στόχος του αλγορίθμου «A priori», είναι να ανιχνεύσει τα συχνά σύνολα στοιχείων στα δεδομένα, δηλαδή τα σύνολα στοιχείων που εμφανίζονται συχνά μαζί.
Η λειτουργία του αλγόριθμου αποτελείται από δύο βασικά βήματα. Αρχικά, βρίσκονται όλα τα συχνά σύνολα στοιχείων, στο σύνολο των δεδομένων. Στη συνέχεια, δημιουργούνται οι κανόνες συσχέτισης από τα συχνά σύνολα στοιχείων, χρησιμοποιώντας τα όρια υποστήριξης (support) και εμπιστοσύνης (confidence).
Ο κανόνας «A priori», επιτρέπει στον αλγόριθμο να περιορίσει τον αριθμό των υποστηριζόμενων συνόλων που χρειάζεται να ψάξει, βελτιώνοντας έτσι την απόδοσή του.

Διαφορές Μάθησης Με Επίβλεψη και Χωρίς Επίβλεψη

Η μεγαλύτερη διάκριση μεταξύ της μάθησης με επίβλεψη και στη μάθηση χωρίς επίβλεψη, είναι ότι ενώ η πρώτη ‘μαθαίνει’ μέσω παραδειγμάτων, δηλαδή επιτελείται εκπαίδευση σε ένα δείγμα δεδομένων, η δεύτερη δεν έχει δάσκαλο, δηλαδή δεν εκπαιδεύεται, αλλά αυτό-διδάσκεται, ανακαλύπτοντας μόνη της ομοιότητες, διαφορές και μοτίβα που οδηγούν σε συμπεράσματα.

Μία ακόμα διάκριση, είναι ότι στη μάθηση με επίβλεψη υπάρχει η γνώση του προκαθορισμένου συνόλου των ετικετών, βάση του οποίου επιτελείται η ομαδοποίηση των δεδομένων, έχει δηλαδή πρόσβαση, σε όλες τις σωστές εξόδους που μπορεί να ανήκουν τα δεδομένα, ενώ στη μάθηση χωρίς επίβλεψη δεν υπάρχει αυτή η γνώση και οι ετικέτες των δεδομένων, πρέπει να βρεθούν από την ίδια την μέθοδο μάθησης, ανακαλύπτοντας ομοιότητες και διαφορές στα δεδομένα, λειτουργώντας ανεξάρτητα.



Σχήμα 18: Διαφορά Μάθησης Με Επίβλεψη και Χωρίς

Η καθεμία από αυτές τις μεθόδους μηχανικής μάθησης, έχει διαφορετικούς στόχους και εφαρμογές, που επίσης τις διαφοροποιούν.

Η μέθοδος της μάθησης με επίβλεψη, έχει ως βάση την εύρεση της γνώσης των σχέσεων, που έχουν τα δεδομένα εισόδου με τα δεδομένα εξόδου. Για παράδειγμα, ένας αλγόριθμος μάθησης με επίβλεψη, μπορεί να χρησιμοποιηθεί για τη πρόβλεψη

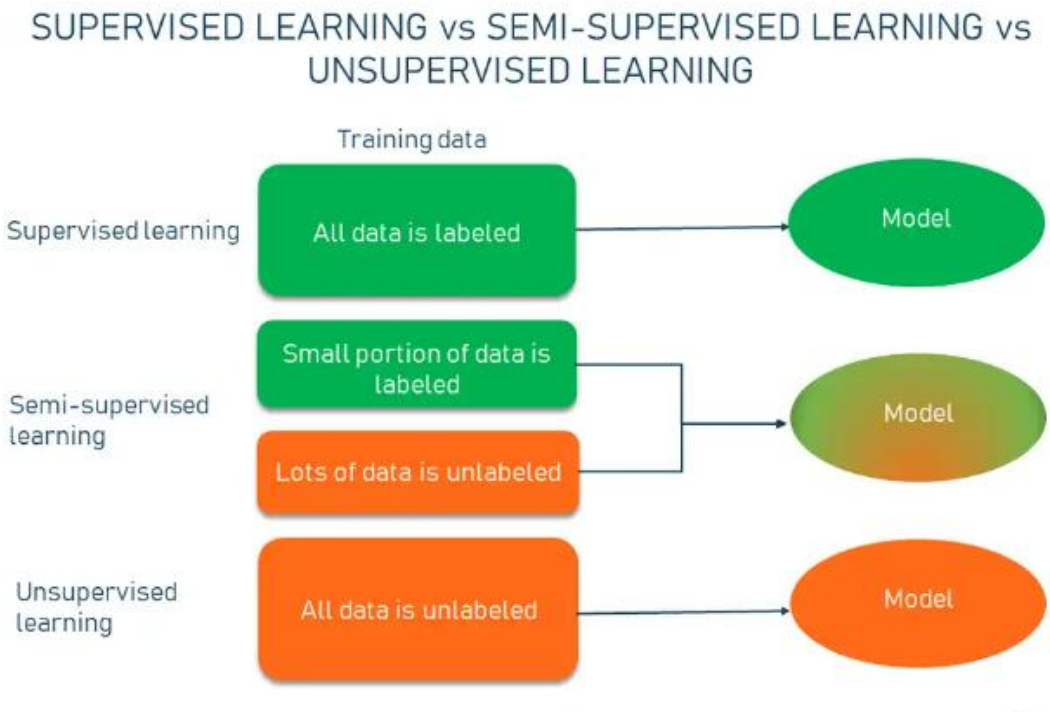
του χρόνου που απαιτείται σε ένα όχημα, να φτάσει στον προορισμό του, βασιζόμενο σε παραμέτρους, όπως η απόσταση, η κίνηση στους δρόμους, οι καιρικές συνθήκες, τυχόν εμπόδια όπως ακινητοποιημένα οχήματα στην διαδρομή κ.α.

Από την άλλη, η μάθηση χωρίς επίβλεψη, έχει ως βάση την ανακάλυψη νέων μοτίβων και σχέσεων σε νέα-ακατέργαστα δεδομένα εισόδου και δεν υπάρχει προκαθορισμένη γνώση των ετικετών, που θα προκύψουν στα δεδομένα εξόδου. Για παράδειγμα, ένας αλγόριθμος μάθησης χωρίς επίβλεψη, μπορεί να χρησιμοποιηθεί για την ανίχνευση μοτίβων σε αγοραστές, με κοινά χαρακτηριστικά, όπως η αγορά σχετικών προϊόντων, για την παροχή διαφημίσεων σε παρόμοιους πελάτες για νέα προϊόντα.

Επομένως γίνεται αντιληπτό, ότι η κάθε μία από τις δύο μεθόδους μηχανικής μάθησης, καλούνται να επιλύσουν διαφορετικούς τύπους προβλημάτων. Η μάθηση με επίβλεψη θεωρείται ότι είναι κατάλληλη για εργασίες ταξινόμησης και παρεμβολής, όπως είναι η πρόβλεψη καιρού, ανάλυση εικόνας και ήχου, καθώς και η ανίχνευση ανεπιθύμητων μηνυμάτων. Ενώ η μάθηση χωρίς επίβλεψη, θεωρείται πιο ιδανική για διερευνητική ανάλυση δεδομένων, όπως ο εντοπισμός ανωμαλιών και θορύβου στα δεδομένα, η οπτικοποίηση μεγάλων δεδομένων, καθώς και η πρόβλεψη αγοράς νέων προϊόντων από πελάτες, βάση ανίχνευσης μοτίβων σε αυτούς.

Η επιλογή λοιπόν ανάμεσα στις μεθόδους μάθησης με επίβλεψη και μάθησης χωρίς επίβλεψη, εξαρτάται από τον τύπο του προβλήματος που καλείται να επιλύσει, το είδος και το μέγεθος των δεδομένων που είναι διαθέσιμα, καθώς και η τεχνογνωσία και η εμπειρία δημιουργίας και διαχείρισης των αλγορίθμων που θα χρειαστούν.

Μπορεί όμως να υπάρξουν περιπτώσεις, που και οι δύο μέθοδοι μηχανικής μάθησης με επίβλεψη και χωρίς επίβλεψη, δεν είναι ικανές να καλύψουν τις ανάγκες επίλυσης. Για αυτές τις περιπτώσεις, υπάρχει και μία τρίτη προσέγγιση που ονομάζεται ημί-εποπτευόμενη μάθηση (Semi-Supervised Learning). Η μέθοδος αυτή, είναι συνδυαστική των άλλων δύο μεθόδων. Οι αλγόριθμοί της, χρησιμοποιούν δεδομένα με προκαθορισμένες ετικέτες, αλλά και ακατέργαστα δεδομένα, που δεν είναι προσημασμένα για να εκπαιδευτούν.



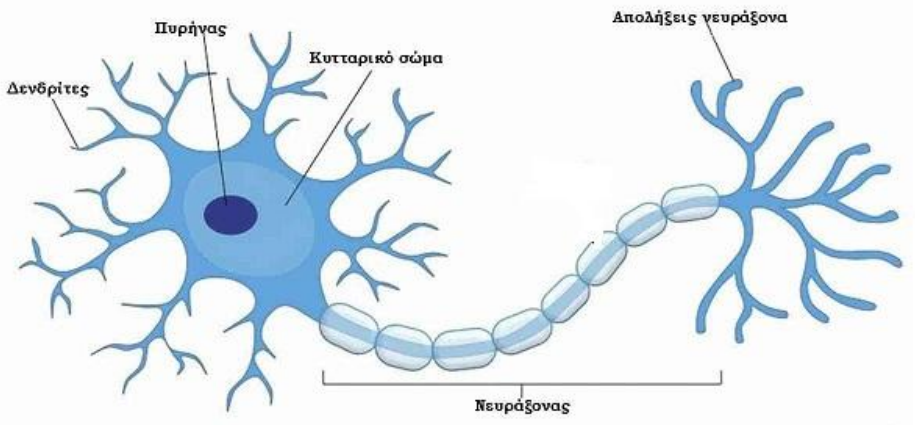
Σχήμα 19: Supervised vs Unsupervised vs Semi-Supervised Learning

Με αυτό τον τρόπο, η μέθοδος της ημί-επιβλεπόμενης μάθησης, λειτουργεί σε μια μεγάλη γκάμα προβλημάτων από κατηγοριοποίηση και παρεμβολή, μέχρι συσταδοποίηση και κανόνες συσχέτισης. Αντιπροσωπευτικά παραδείγματα ημί-εποπτευόμενης μάθησης, είναι η αναγνώριση εικόνας και ομιλίας, κατηγοριοποίηση περιεχόμενου εγγράφων κειμένου κ.α.

ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

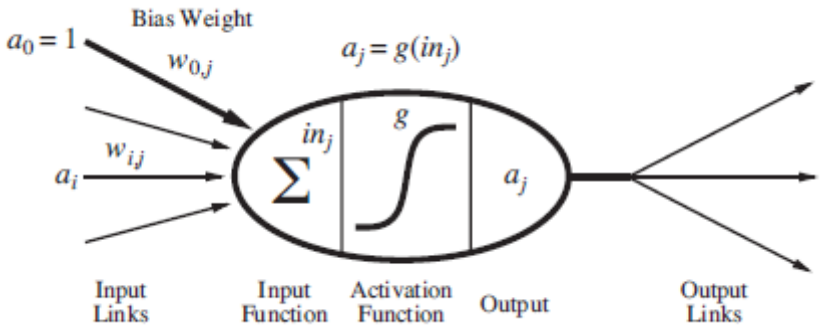
Εισαγωγή

Στον εγκέφαλο του ανθρώπου, η νοητική δραστηριότητα συνίσταται κυρίως από ηλεκτροχημική δραστηριότητα, σε δίκτυα εγκεφαλικών κυττάρων, που ονομάζονται νευρώνες (Σχήμα 20). Ο κάθε νευρώνας, αποτελείται από ένα κυτταρικό σώμα, το οποίο έχει έναν πυρήνα. Οι απολήξεις του κυτταρικού σώματος, ονομάζονται συνάψεις. Οι συνάψεις, έχουν το ρόλο της εισόδου ενός νευρώνα. Από την κάθε σύναψη, εκτείνονται πολλαπλές ίνες, που ονομάζονται δενδρίτες. Σε κάθε δενδρίτη, εισέρχεται ένα πάρα πολύ μικρό έντασης ηλεκτρικό σήμα, το οποίο οδηγείται προς το κυτταρικό σώμα μέσω των συνάψεων, οι οποίες έχουν τον επιπλέον ρόλο της μεταβολής του διερχόμενου ηλεκτρικού ρεύματος, αυξομειώνοντάς το. Στον πυρήνα, αφού φτάσει το ηλεκτρικό ρεύμα από όλες τις συνάψεις, διενεργείται μια πολύπλοκη ηλεκτροχημική διεργασία, με αποτέλεσμα την δημιουργία ενός νέου ηλεκτρικού ρεύματος, ως συνάρτηση του σήματος εισόδου. Από το κυτταρικό σώμα, εκτείνεται μια μακριά ίνα, που ονομάζεται άξονας. Στον άξονα, καταλήγει το ηλεκτρικό ρεύμα που παράχθηκε στον πυρήνα και αποτελεί την έξοδο του νευρώνα. Ο κάθε νευρώνας συνδέεται με 10 έως 100.000 άλλους νευρώνες, σε πολλαπλές διασταυρώσεις. Η διασύνδεση πολλαπλών νευρώνων, δημιουργεί ένα νευρωνικό δίκτυο μεταφοράς ηλεκτρικών σημάτων. Αυτοί οι μηχανισμοί, θεωρείται ότι διέπουν τη μάθηση στον εγκέφαλο.



Σχήμα 20: Τα μέρη του βιολογικού νευρώνα

Εμπνευσμένη από τον βιολογικό νευρώνα, κάποιες πρώτες έρευνες τεχνητής νοημοσύνης, είχαν ως στόχο τη δημιουργία τεχνητού νευρώνα. Στο Σχήμα 21 παρουσιάζεται ένα απλό μαθηματικό μοντέλο ενός τεχνητού νευρώνα, που αναπτύχθηκε από τους McCulloch και Pitts (1943). Κάθε τεχνητός νευρώνας, έχει μια ή πολλαπλές εισόδους (inputs) και παράγει μια έξοδο (output), που μπορεί να σταλεί ως είσοδος, σε πολλούς άλλους νευρώνες. Οι είσοδοι συμβολίζονται με α_1, α_2, α_3 … α_i και μπορούν να πάρουν οποιεσδήποτε τιμές δεδομένων, όπως οι τιμές χαρακτηριστικών ενός εξωτερικού δείγματος δεδομένων ή μπορεί να είναι η έξοδος ενός άλλου νευρώνα. Με w_1, w_2, w_3 … w_i συμβολίζουμε τα βάρη των συνάψεων των εισόδων, του τεχνητού νευρώνα. Υπάρχει μια σταθερή ειδική είσοδος που συμβολίζεται με α_0 και αντίστοιχα το βάρος της με w_0. Διακρίνονται δύο περιπτώσεις για την ειδική είσοδο και το βάρος της. Η πρώτη ονομάζεται κατώφλι (threshold), όπου $\alpha_0 = -1$ και $w_0 = \theta$, και η δεύτερη που ονομάζεται πόλωση (bias), όπου $\alpha_0 = 1$ και $w_0 = b$. Για να βρούμε την έξοδο ενός νευρώνα, λαμβάνουμε το άθροισμα όλων των εισόδων, σταθμισμένο με τα βάρη των συνάψεων από τις εισόδους στο νευρώνα (Input Function). Αυτό το σταθμισμένο άθροισμα, αναφέρεται μερικές φορές ως ενεργοποίηση. Το σταθμισμένο άθροισμα, περνάει μέσα από μια (συνήθως μη γραμμική) συνάρτηση ενεργοποίησης (Activation Function), για να παραχθεί η έξοδος.



Σχήμα 21: Απλό μαθηματικό μοντέλο Τεχνητού Νευρώνα

Οι βασικές συναρτήσεις ενεργοποίησης είναι οι εξής:

1. *Βηματική συνάρτηση ή συνάρτηση κατωφλίου*

   Ονομάζεται και απλός αισθητήρας ή perception. Οι τιμές της περιορίζονται πάντα στο διάστημα (0,1), για οποιαδήποτε τιμή της εισόδου. Επειδή έχει το

μειονέκτημα, ότι η παράγωγός της είναι μηδέν και επειδή η μάθηση στα ΤΝΔ εκτελείται με τη μεταβολή των τιμών των βαρών, σχετιζόμενη με την παράγωγο, δεν θεωρείται βολική ως συνάρτηση ενεργοποίησης.

2. *Συνάρτηση πρόσημου*

   Ίδια με την βηματική συνάρτηση, με διαφοροποίηση ότι οι τιμές της περιορίζονται πάντα στο διάστημα (-1,1).

3. *Γραμμική συνάρτηση*

   Η τελική έξοδος / ενεργοποίηση του νευρώνα, είναι ευθέως ανάλογη της εισόδου του, ή ίση μαζί της. Χρήσιμη, όταν ο στόχος είναι η μάθηση ενός προβλήματος παρεμβολής.

4. *Λογιστική / Σιγμοειδής συνάρτηση*

   Η σιγμοειδής συνάρτηση, είναι από τις πιο βασικές στην κατασκευή ενός ΤΝΔ. Οι τιμές της περιορίζονται πάντα στο διάστημα (0,1), για οποιαδήποτε τιμή της εισόδου (για αυτό ονομάζεται και συμπιέζουσα συνάρτηση). Έτσι, εξασφαλίζεται ότι η έξοδος δεν παίρνει μεγάλες τιμές (δεν απειρίζεται). Συμπεριφέρεται καλά ,για όλα τα μεγέθη τιμών και είναι παντού συνεχής και παραγωγίσιμη.

Αναλόγως των βιολογικών νευρωνικών δικτύων, η διασύνδεση των τεχνητών νευρώνων μεταξύ τους, δημιουργεί ένα τεχνητό νευρωνικό δίκτυο (ΤΝΔ).[2] [3]

Αρχιτεκτονική Τεχνητών Νευρωνικών Δικτύων

Το τεχνητό νευρωνικό δίκτυο (ΤΝΔ), είναι μια δομή συνδεδεμένων τεχνητών νευρώνων, με σκοπό την κατάλληλη επεξεργασία πληροφοριών και είναι εμπνευσμένο από το βιολογικό νευρωνικό δίκτυο του ανθρώπινου εγκεφάλου. Το κάθε ΤΝΔ, κατασκευάζεται αναλόγως την λειτουργία που καλείται να εκτελέσει, όπως η αναγνώριση μοτίβων ή η κατηγοριοποίηση δεδομένων, μέσω της διαδικασίας της μάθησης. Η διαδικασία της μάθησης, επιτελείται από την επιστήμη της μηχανικής μάθησης. Τα ΤΝΔ, αποτελούν δηλαδή ένα τύπο μοντέλου της μηχανικής μάθησης, η οποία αναλύθηκε σε προηγούμενα κεφάλαια.

Όπως προαναφέρθηκε, ένα ΤΝΔ, είναι μια δομή από συνδεδεμένους μεταξύ τους τεχνητούς νευρώνες, οι οποίοι έχουν ένα σχετικό βάρος και κατώφλι και είναι

διατεταγμένοι σε μια ακολουθία από διαφορετικά επίπεδα. Η έξοδος του κάθε επιπέδου, αποτελεί την είσοδο στο επόμενο επίπεδο. Ένας νευρώνας ενός επιπέδου, μπορεί να συνδεθεί με όλους (fully connected) ή ένα υποσύνολο (partially connected), των νευρώνων του επόμενου επιπέδου. Ένα ΤΝΔ, αποτελείται από τρία διαφορετικά επίπεδα:

- *Επίπεδο Εισόδου (Input Layer)*

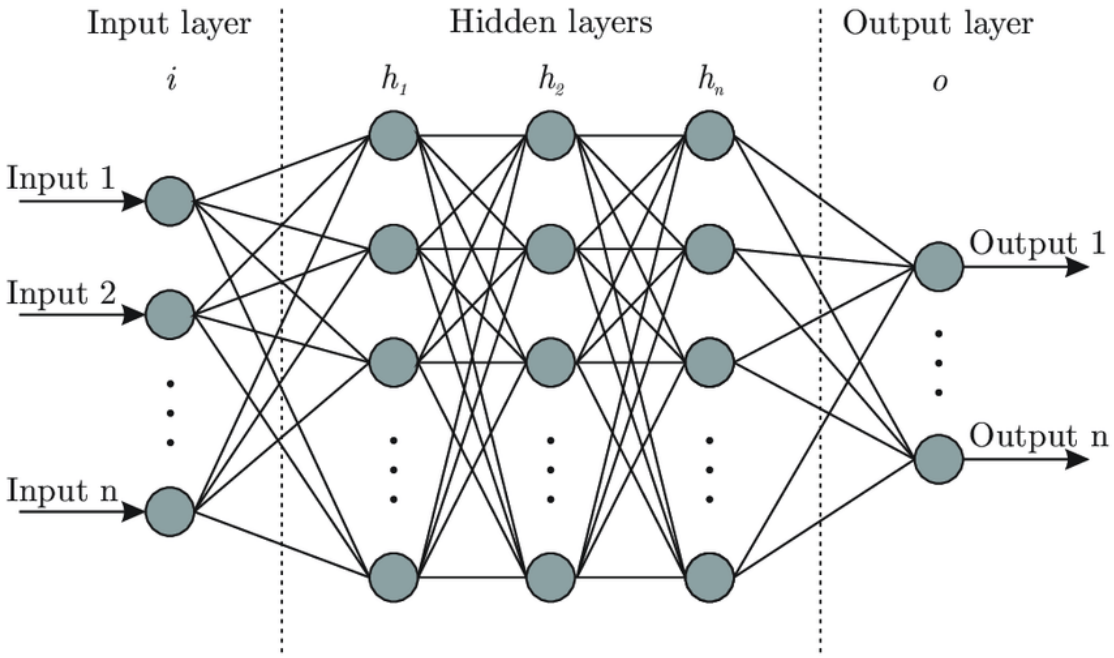
  Αποτελεί την είσοδο του ΤΝΔ και η οποία τροφοδοτείται από τα δεδομένα εισόδου. Το επίπεδο εισόδου είναι μοναδικό και αποτελείται από έναν ή περισσότερους νευρώνες, τόσους όσους και τα χαρακτηριστικά-γνωρίσματα των δεδομένων εισόδου.

- *Κρυφό Επίπεδο (Hidden Layer)*

  Το κρυφό επίπεδο είναι το ενδιάμεσο επίπεδο ανάμεσα στο επίπεδο εισόδου και το επίπεδο εξόδου. Λαμβάνει τα ακατέργαστα δεδομένα από το επίπεδο εισόδου και είναι υπεύθυνο για την κατάλληλη επεξεργασία τους, προωθώντας τα επεξεργασμένα δεδομένα στο επίπεδο εξόδου. Το κρυφό επίπεδο μπορεί να μην υπάρχει, να είναι μοναδικό ή να υπάρχουν πολλαπλά κρυφά επίπεδα, τα οποία συνδέονται μεταξύ τους. Σε αυτή την περίπτωση, το πρώτο κρυφό επίπεδο, λαμβάνει τα δεδομένα από το επίπεδο εισόδου ως είσοδο, επιτελεί την επεξεργασία για την οποία είναι υπεύθυνο και τροφοδοτεί με τα μετασχηματισμένα αποτελέσματα το επόμενο κρυφό επίπεδο. Το τελευταίο κρυφό επίπεδο, τροφοδοτεί με τα τελικά αποτελέσματα το επίπεδο εξόδου.

- *Επίπεδο Εξόδου (Output Layer)*

  Λαμβάνει τα επεξεργασμένα δεδομένα από το κρυφό επίπεδο, επιτελεί με την σειρά του την κατάλληλη επεξεργασία και δίνει την έξοδο με τα τελικά δεδομένα που έπρεπε να επιτελέσει το ΤΝΔ. Το επίπεδο αυτό είναι μοναδικό.



Σχήμα 22: Αρχιτεκτονική Τεχνητού Νευρωνικού Δικτύου

Το κάθε ΤΝΔ, δεν μπορεί να αξιοποιηθεί για να επιλύσει πολλαπλά και διαφορετικά προβλήματα. Αναλόγως το πρόβλημα επιλέγεται και ο κατάλληλος τύπος αρχιτεκτονικής ΤΝΔ, δημιουργούνται τα απαραίτητα επίπεδα και παραμετροποιείται αναλόγως αναγκών. Τα ΤΝΔ, χρησιμοποιούνται και στις δύο μεθόδους μηχανικής μάθησης με επίβλεψη και χωρίς επίβλεψη, για την κάλυψη διαφορετικών κατηγοριών προβλημάτων όπως:

- Πρόβλεψη (π.χ. ισοτιμίας νομισμάτων ή τιμών μετοχών, καιρού)
- Κατηγοριοποίηση (π.χ. κατηγοριοποίηση ιατρικών εικόνων, εικόνων από ραντάρ, πελατών βάση αγοραστικών τους συνηθειών)
- Αναγνώριση (π.χ. προσώπου σε συστήματα ασφάλειας, εικόνας, ήχου, κειμένου)
- Αποτίμηση (π.χ. παρακολούθηση στόχων, εντοπισμός κίνησης, ταύτιση δακτυλικών αποτυπωμάτων, έλεγχος προϊόντων)[1] [6] [9]

Λειτουργία Τεχνητού Νευρωνικού Δικτύου

Τα ΤΝΔ, αποτελούνται συνήθως από τρία επίπεδα τεχνητών νευρώνων. Κάθε επίπεδο, λαμβάνει είσοδο από το προηγούμενο επίπεδο ή από τα δεδομένα εισόδου και

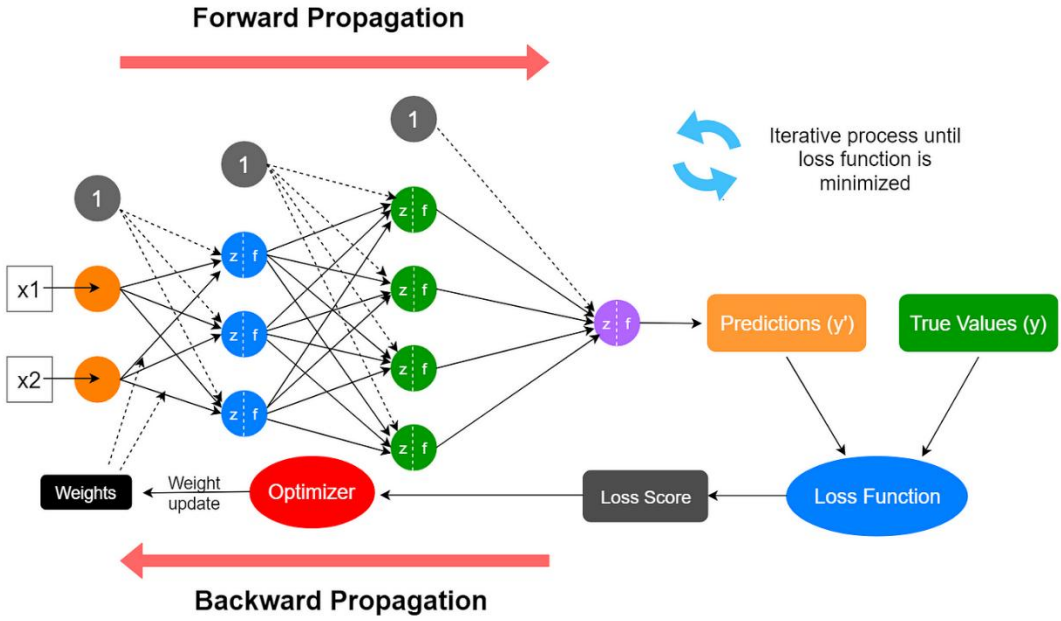
εφαρμόζει κάποιους μαθηματικούς μετασχηματισμούς, για να παράγει μια έξοδο. Η επεξεργασία αυτή επαναλαμβάνεται, μέχρι η έξοδος να φτάσει στο τελικό επίπεδο, το οποίο παράγει το τελικό αποτέλεσμα.

Το μοντέλο ενός ΤΝΔ, μπορεί να καθοριστεί από τις εξής οντότητες:

- *Τεχνητοί Νευρώνες (Artificial Neurons):* Οι τεχνητοί νευρώνες στο ΤΝΔ ονομάζονται και κόμβοι και είναι οι βασικές μονάδες επεξεργασίας στα ΤΝΔ. Κάθε νευρώνας, λαμβάνει είσοδο από πολλούς άλλους νευρώνες ή από τα δεδομένα εισόδου, επεξεργάζεται αυτή την είσοδο, χρησιμοποιώντας ένα σχετικό σύνολο βαρών και μια συνάρτηση ενεργοποίησης και παράγει ένα αποτέλεσμα. Για τους τεχνητούς νευρώνες, αναφερθήκαμε σε προηγούμενη ενότητα πιο αναλυτικά.
- *Βάρη (Weights):* Τα βάρη, αντιπροσωπεύουν τη δύναμη της σύνδεσης μεταξύ των νευρώνων. Κατά την εκπαίδευση, τα βάρη προσαρμόζονται, ώστε το νευρωνικό δίκτυο, να μπορεί να μάθει να αναγνωρίζει πρότυπα στα δεδομένα εισόδου.
- *Συνάρτηση Ενεργοποίησης (Activation Function):* Η συνάρτηση ενεργοποίησης, καθορίζει το επίπεδο δραστηριότητας του νευρώνα, βάσει της συνολικής εισόδου που λαμβάνει. Μερικές συνηθισμένες συναρτήσεις ενεργοποίησης, περιλαμβάνουν τη σιγμοειδή και την επίπεδη γραμμή.
- *Επίπεδα (Layers):* Τα ΤΝΔ αποτελούνται από διαδοχικά επίπεδα νευρώνων. Το πρώτο επίπεδο, λαμβάνει την αρχική είσοδο και ονομάζεται επίπεδο εισόδου, ενώ το τελευταίο επίπεδο, παράγει την τελική έξοδο του δικτύου και ονομάζεται επίπεδο εξόδου. Τα ενδιάμεσα επίπεδα ονομάζονται κρυφά επίπεδα και εκτελούν την ενδιάμεση επεξεργασία των δεδομένων. Στην προηγούμενη ενότητα αναφερθήκαμε πιο αναλυτικά για τα επίπεδα.
- *Συνδέσεις (Connections):* Οι τεχνητοί νευρώνες σε διαδοχικά επίπεδα, συνδέονται μεταξύ τους μέσω συνδέσεων. Κάθε σύνδεση, έχει ένα βάρος που καθορίζει τη σημασία της επικοινωνίας, μεταξύ των δύο νευρώνων.
- *Εκπαίδευση (Training):* Κατά τη διάρκεια της εκπαίδευσης, το ΤΝΔ προσαρμόζει τα βάρη του, ώστε να μπορεί να παράγει τη σωστή έξοδο για μια δεδομένη είσοδο. Αυτό επιτυγχάνεται συνήθως, μέσω μιας διαδικασίας που ονομάζεται ανάδραση ή αλγόριθμος βελτιστοποίησης, όπως η ανάπτυξη προς τα πίσω (backpropagation).

Εκπαίδευση Τεχνητού Νευρωνικού Δικτύου

Η εκπαίδευση ενός ΤΝΔ, είναι η διαδικασία κατά την οποία, το δίκτυο προσαρμόζει τα βάρη του μέσω εκθέσεων σε δεδομένα εκπαίδευσης, ώστε να εκπαιδευτεί να παράγει την επιθυμητή έξοδο, για μια δεδομένη είσοδο. Η διαδικασία αυτή, απαιτεί τη χρήση ενός αλγορίθμου βελτιστοποίησης (optimization algorithm), για να προσαρμόσει τα βάρη που παράγει το δίκτυο στην έξοδο, σε σχέση με τις πραγματικές τιμές. Ο υπολογισμός του σφάλματος, δηλαδή της απόκλισης των δεδομένων εξόδου με τα δεδομένα εισόδου, επιτυγχάνεται με την κατάλληλη συνάρτηση κόστους (loss function).



Σχήμα 23: Διαδικασία Εκπαίδευσης ΤΝΔ

Η κύρια διαδικασία εκπαίδευσης ενός ΤΝΔ, συνήθως περιλαμβάνει τα ακόλουθα βήματα:

1. *Ορισμός του Προβλήματος:* Καθορισμός του είδους του προβλήματος που πρέπει το ΤΝΔ να λύσει (π.χ. αναγνώριση εικόνων, κατηγοριοποίηση δεδομένων, πρόβλεψη χρονοσειρών κ.λπ.).

2. *Συλλογή Δεδομένων:* Συλλογή δεδομένων που περιλαμβάνουν είσοδο και την αντίστοιχη επιθυμητή έξοδο. Αυτά τα δεδομένα, χρησιμοποιούνται για την εκπαίδευση του δικτύου (test data).
3. *Ορισμός της Αρχιτεκτονικής του Δικτύου:* Περιλαμβάνει τον καθορισμό του αριθμού των επιπέδων του δικτύου, του αριθμού των νευρώνων σε κάθε επίπεδο, των συναρτήσεων ενεργοποίησης κ.λπ.
4. *Αρχικοποίηση των Βαρών:* Αρχική ανάθεση τυχαίων τιμών στα βάρη του δικτύου.
5. *Επαναλαμβανόμενη Εκπαίδευση:* Το δίκτυο εκπαιδεύεται επαναλαμβανόμενα επί διαδοχικών εποχών δεδομένων εκπαίδευσης. Κατά τη διάρκεια κάθε εποχής, τα δεδομένα εισόδου περνούν μέσα από το δίκτυο, υπολογίζεται το σφάλμα εξόδου και το σφάλμα αυτό τροφοδοτείται ξανά στο ΤΝΔ, για την προσαρμογή των βαρών, μέσω της συνάρτησης βελτιστοποίησης.
6. *Αξιολόγηση:* Κατά τη διάρκεια και μετά την εκπαίδευση, το δίκτυο αξιολογείται χρησιμοποιώντας ένα σύνολο δεδομένων ελέγχου (check data). Αυτό γίνεται, για να ελεγχθεί η απόδοση του δικτύου σε νέα δεδομένα, που δεν είχαν χρησιμοποιηθεί κατά την εκπαίδευση και να αποφευχθεί η υπέρ-προσαρμογή.
7. *Προσαρμογή των Παραμέτρων:* Ανάλογα με την απόδοση του δικτύου στο σύνολο ελέγχου, οι παράμετροι του δικτύου μπορεί να ρυθμιστούν εκ νέου ή να βελτιωθούν για βέλτιστη απόδοση.

Αυτά τα βήματα, επαναλαμβάνονται μέχρι να επιτευχθεί η επιθυμητή απόδοση του δικτύου, στο πρόβλημα που επιδιώκει να λύσει. Η διαδικασία της εκπαίδευσης, απαιτεί συνήθως μεγάλο όγκο υπολογισμών και δεδομένων και η αποτελεσματική εκπαίδευση ενός ΤΝΔ, μπορεί να απαιτήσει υψηλή υπολογιστική ισχύ και πόρους.[1] [6]

Τύποι Αρχιτεκτονικής Τεχνητών Νευρωνικών Δικτύων

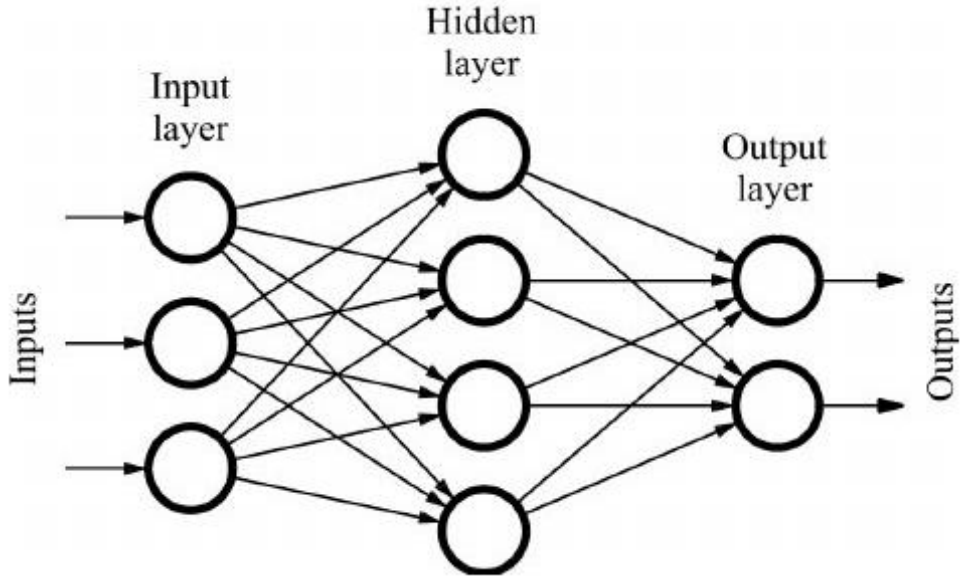
Υπάρχουν πολλοί και διαφορετικοί τύποι αρχιτεκτονικής για τα ΤΝΔ, που ο κάθε ένας επιτελεί διαφορετική λειτουργία, ανάλογη του προβλήματος που καλείται να επιλύσει.

Οι πιο βασικοί τύποι αρχιτεκτονικής ΤΝΔ είναι οι εξής:

- Νευρωνικά Δίκτυα Πρόσθιας Τροφοδότησης (Feedforward Neural Networks - FNN)
- Νευρωνικά Δίκτυα Ανατροφοδότησης (Recurrent Neural Networks – RNN)
- Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks – CNN)
- Αυτόματοι Κωδικοποιητές (Autoencoders)

ΤΝΔ Πρόσθιας Τροφοδότησης (Feedforward Neural Networks - FNN)

Αυτός είναι ο απλούστερος τύπος αρχιτεκτονικής ΤΝΔ, όπου οι πληροφορίες ρέουν προς μία κατεύθυνση, από την είσοδο στην έξοδο. Τα επίπεδα, είναι πλήρως συνδεδεμένα (fully connected), δηλαδή κάθε νευρώνας σε ένα επίπεδο, συνδέεται με όλους τους νευρώνες στο επόμενο επίπεδο. Μπορεί να έχει κανένα, ένα ή πολλά κρυφά επίπεδα. Ο αριθμός των νευρώνων των επιπέδων εισόδου και εξόδου, εκτιμώνται με βάση τα δεδομένα της εισόδου και τα δεδομένα της εξόδου.



Σχήμα 24: Παράδειγμα ΤΝΔ Εμπρόσθιας Τροφοδότησης (με ένα κρυφό επίπεδο)

Η εκπαίδευση του FNN συντελείται μέσω δύο σταδίων: το στάδιο της εμπρόσθιας τροφοδότησης και το στάδιο της οπισθοδιάδοσης.

Στάδιο εμπρόσθιας τροφοδότησης:

Σε αυτό το στάδιο, τα δεδομένα εισόδου τροφοδοτούνται στο δίκτυο και διαδίδονται μόνο προς τα εμπρός. Το κάθε κρυφό επίπεδο, λαμβάνει τα δεδομένα αυτά, αφού υπολογιστεί το σταθμισμένο άθροισμά τους και επεξεργαστούν από μια συνάρτηση ενεργοποίησης. Η διαδικασία αυτού του σταδίου, ολοκληρώνεται όταν τα δεδομένα περάσουν από όλα τα τυχόν κρυφά επίπεδα και τα τελικά μετασχηματισμένα δεδομένα φτάσουν στο επίπεδο εξόδου, που θα παράγει μια πρόβλεψη.

Στάδιο οπισθοδιάδοσης:

Αφού επιτευχθεί μια πρόβλεψη, γίνεται υπολογισμός του σφάλματος, το οποίο με την σειρά του διαδίδεται ξανά στο δίκτυο. Τα βάρη των συνδέσεων των επιπέδων, προσαρμόζονται με χρήση κατάλληλου αλγορίθμου, με σκοπό την ελαχιστοποίηση του σφάλματος αυτού.

Τα δύο στάδια συνεχίζονται, έως ότου η εκπαίδευση του FNN ολοκληρωθεί. Η εκπαίδευση του FNN ολοκληρώνεται, όταν αποδοθούν τα δεδομένα εκπαίδευσης και τα βάρη έχουν προσαρμοστεί, έτσι ώστε να έχει ελαχιστοποιηθεί το σφάλμα.

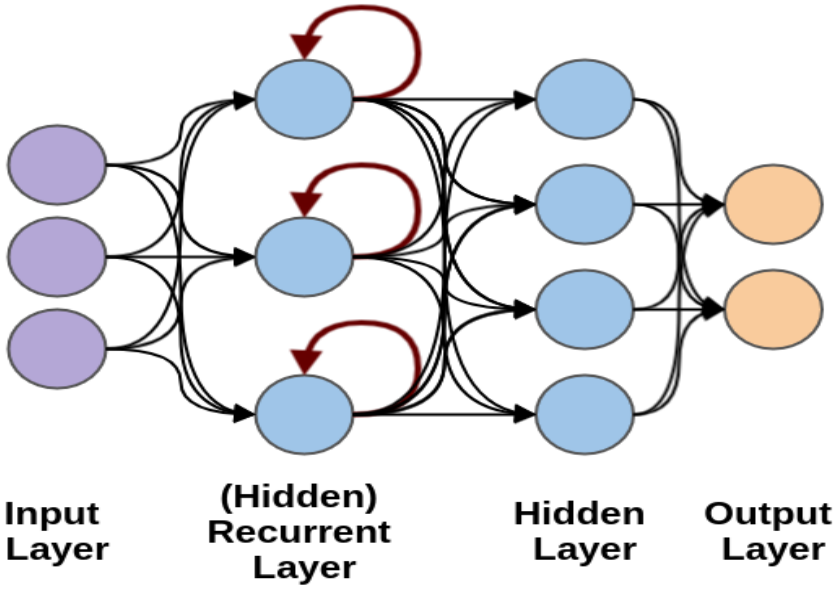
Σημαντικό είναι να είναι σωστή η επιλογή του αριθμού των κρυφών επιπέδων, καθώς και ο αριθμός των νευρώνων του κάθε επιπέδου, έτσι ώστε να βελτιστοποιηθεί η απόδοση του FNN. Επίσης, είναι επιρρεπές στην υπέρ-προσαρμογή.

Τα FNN, χρησιμοποιούνται στην μηχανική μάθηση με επίβλεψη.[1] [9]

ΤΝΔ Ανατροφοδότησης (Recurrent Neural Networks – RNN)

Αυτά τα δίκτυα έχουν ένα στοιχείο "μνήμης", όπου οι πληροφορίες μπορούν να ρέουν σε κύκλους, μέσω του δικτύου. Για παράδειγμα, στην περίπτωση πρόβλεψης της επόμενης λέξης μιας πρότασης, θα έπρεπε να ‘θυμάται’ το σύστημα τις προηγούμενες λέξεις, δηλαδή να υπάρχει η ικανότητα της μνήμης. Το πρόβλημα αυτό, επιλύθηκε με τα RNN, με την βοήθεια μιας κρυφής κατάστασης, που ονομάζεται κατάσταση μνήμης, η οποία θυμάται πληροφορίες για μια ακολουθία, επιτρέποντας στο δίκτυο

να καταγράφει διαδοχικές εξαρτήσεις, απομνημονεύοντας προηγούμενες εισόδους κατά την επεξεργασία.



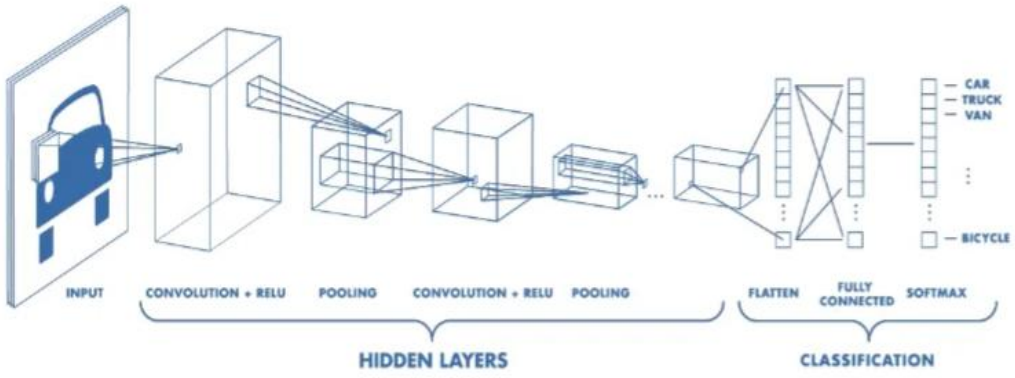
Σχήμα 25: Παράδειγμα ΤΝΔ Ανατροφοδότησης

Η αρχιτεκτονική επιπέδων εισόδου και εξόδου, είναι ίδια με τους άλλους τύπους αρχιτεκτονικής ΤΝΔ. Η διαφορά τους έγκειται, στον τρόπο με τον οποίο οι πληροφορίες ρέουν από την είσοδο στην έξοδο. Αποτελείται από πολλαπλές σταθερές μονάδες λειτουργίας ενεργοποίησης, μία για κάθε χρονικό βήμα. Κάθε μονάδα, έχει μια κρυφή κατάσταση, η οποία υποδηλώνει την προηγούμενη γνώση που διατηρεί το δίκτυο αυτήν τη στιγμή, σε ένα δεδομένο χρονικό βήμα. Αυτή η κρυφή κατάσταση, ενημερώνεται σε κάθε βήμα για να υποδηλώσει την αλλαγή στη γνώση του δικτύου, για το παρελθόν. Επομένως, σε κάθε χρονικό βήμα στο RNN, υπολογίζεται η τρέχουσα κατάστασή του, λαμβάνοντας υπόψη την τρέχουσα είσοδο δεδομένων και την προηγούμενη κατάστασή του. Η τρέχουσα κατάσταση, γίνεται η προηγούμενη κατάσταση του επόμενου χρονικού βήματος. Τα χρονικά βήματα που επιτελούνται, είναι ανάλογα του προβλήματος που καλείται να επιλύσει το RNN, ενώνοντας τις πληροφορίες από όλες τα προηγούμενες καταστάσεις. Μετά την ολοκλήρωση των χρονικών βημάτων, η τελική τρέχουσα κατάσταση, αποτελεί το αποτέλεσμα για τον υπολογισμό της εξόδου και του σφάλματος. Το σφάλμα διοχετεύεται ξανά στο RNN (οπισθοδιάδοση), για την κατάλληλη ενημέρωση των βαρών και επανάληψη της

διαδικασίας, έως ότου το σφάλμα ελαχιστοποιηθεί και η απόδοση του RNN είναι ικανοποιητική, τερματίζοντας με αυτό τον τρόπο την εκπαίδευσή του.
Το μεγάλο πλεονέκτημα λοιπόν του RNN, είναι η ικανότητά του να θυμάται κάθε πληροφορία μέσα στο χρόνο, καθιστώντας το ιδανικό για την επεξεργασία αλληλουχίας δεδομένων, όπως πρόβλεψη χρονοσειρών, αναγνώριση ομιλίας, μετάφραση, αναγνώριση εικόνας, ανίχνευση προσώπου ή δημιουργία κειμένου.
Τα RNN, χρησιμοποιούνται στην μηχανική μάθηση με επίβλεψη [1] [9]

ΤΝΔ Συνέλιξης (Convolutional Neural Networks – CNN)

Τα CNN έχουν σχεδιαστεί, για να επεξεργάζονται δεδομένα με τοπολογία τύπου πλέγματος, όπως είναι οι εικόνες. Μια ψηφιακή εικόνα, αποτελείται από pixel διατεταγμένα σε μορφή πλέγματος, τα οποία αναπαριστούν χαρακτηριστικά, όπως φωτεινότητα και χρώμα. Τα επίπεδα του CNN, είναι διατεταγμένα με τέτοιο τρόπο, ώστε να πραγματοποιούν ανίχνευση αρχικά σε απλά μοτίβα όπως γραμμές και καμπύλες και στη συνέχεια σε πιο σύνθετα, όπως αντικείμενα και πρόσωπα.
Ένα CNN αποτελείται συνήθως από τρία επίπεδα: ένα επίπεδο συνέλιξης (convolutional layer), ένα επίπεδο συγκέντρωσης (pooling layer) και ένα πλήρως συνδεδεμένο επίπεδο (fully connected layer).



Σχήμα 26: Παράδειγμα ΤΝΔ Συνέλιξης

Επίπεδο Συνέλιξης (Convolutional Layer)

Το επίπεδο συνέλιξης, είναι το πιο βασικό του CNN και επωμίζεται το περισσότερο φορτίο υπολογιστικής ισχύος του CNN. Στην ουσία, εκτελεί πολλαπλασιασμό δύο πινάκων, από τους οποίους ο ένας αποτελείται από το σύνολο των παραμέτρων εκμάθησης (ονομάζεται πυρήνας) και ο άλλος από τα περιορισμένα δεδομένα των pixel της εικόνας. Η επιτυχία της συγκεκριμένης τεχνικής, έγκειται στο γεγονός, ότι ενώ μια εικόνα μπορεί να αποτελείται από εκατομμύρια pixel, κατά την επεξεργασία της χρησιμοποιώντας τον πυρήνα, γίνεται ανίχνευση μόνο σημαντικών πληροφοριών της τάξεως των εκατοντάδων pixel, μειώνοντας με αυτό τον τρόπο τις απαιτήσεις μνήμης του μοντέλου και βελτιώνοντας τη στατιστική απόδοση του.

Επίπεδο Συγκέντρωσης (Pooling Layer)

Στο επίπεδο συγκέντρωσης, επιτελείται αντικατάσταση της εξόδου του CNN, με μια συγκεντρωτική στατιστική των κοντινών εξόδων. Με αυτό τον τρόπο, πραγματοποιείται μείωση του χωρικού μεγέθους της αναπαράστασης, γεγονός που καθιστά την ποσότητα υπολογισμού και βαρών αισθητά ελαχιστοποιημένη. Η λειτουργία αυτή, πραγματοποιείται μέσω κατάλληλης συνάρτησης συγκέντρωσης.

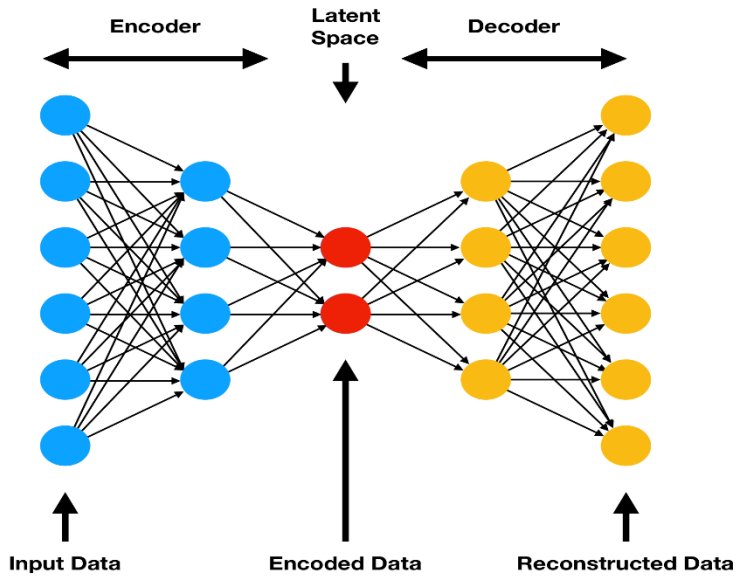
Πλήρως Συνδεδεμένο Επίπεδο (Fully Connected Layer)

Το επίπεδο αυτό, χρησιμοποιείται στη χαρτογράφηση της αναπαράστασης μεταξύ της εισόδου και της εξόδου. Οι νευρώνες που το αποτελούν, έχουν πλήρη συνδεσιμότητα με όλους τους νευρώνες στο προηγούμενο και στο επόμενο επίπεδο. Ο υπολογισμός του, γίνεται με πολλαπλασιασμό πίνακα, ακολουθούμενο από τον επηρεασμό του bias.

Επειδή τα CNN, αξιοποιούν αρχές από τη γραμμική άλγεβρα, όπως τον πολλαπλασιασμό πινάκων για τον εντοπισμό μοτίβων μέσα σε μια εικόνα, μπορεί να είναι απαιτητικά σε υπολογιστική ισχύ, για την εκπαίδευση τους.
Τα CNN, επιτελούν συνήθως εργασίες όρασης υπολογιστή, όπως αναγνώριση προσώπων και αντικειμένων.
Τα CNN, χρησιμοποιούνται στην μηχανική μάθηση με επίβλεψη.[10]

Αυτόματοι Κωδικοποιητές (Autoencoders)

Οι αυτόματοι κωδικοποιητές, είναι σχεδιασμένοι για να επιτελούν συμπίεση των δεδομένων εισόδου, σε έναν κώδικα χαμηλότερης αναπαράστασης και στην συνέχεια ανακατασκευάζουν τα αρχικά δεδομένα στην έξοδο, κάνοντας χρήση του κώδικα αυτού. Ο κώδικας, είναι μια συμπιεσμένη έκδοση των δεδομένων εισόδου.



Σχήμα 27: Παράδειγμα Autoencoder

Ένας αυτόματος κωδικοποιητής, αποτελείται από 3 στοιχεία: τον κωδικοποιητή, τον κώδικα και τον αποκωδικοποιητή. Ο κωδικοποιητής, επιτελεί την συμπίεση της εισόδου παράγοντας τον κωδικό, χρησιμοποιώντας μια μέθοδο κωδικοποίησης και ο αποκωδικοποιητής ανακατασκευάζει την είσοδο κάνοντας χρήση του κωδικού, χρησιμοποιώντας με την σειρά του μια μέθοδο αποκωδικοποίησης. Οι αυτόματοι κωδικοποιητές, κάνουν χρήση και μιας συνάρτησης απώλειας, η οποία συντελεί στην σύγκριση της εξόδου, με την είσοδο και την εύρεση σφάλματος.

Η έξοδος που θα παραχθεί από τον αυτόματο κωδικοποιητή, δεν θα είναι ακριβώς ίδια με την είσοδο, αλλά μια κοντινή υποβαθμισμένη αναπαράσταση της. Δεν μπορούν να χρησιμοποιηθούν για εφαρμογές γενικής χρήσης, αλλά μόνο ειδικής, καθότι μπορούν να αποδώσουν με παρόμοια δεδομένα. Αν δηλαδή ένας autoencoder, είναι εκπαιδευμένος σε χειρόγραφα ψηφία, δεν μπορεί να αποδώσει σε συμπίεση φωτογραφιών τοπίων.

Λόγω των ικανοτήτων τους, οι autoencoders βρίσκουν εφαρμογές σε εργασίες, όπως συμπίεση εικόνας, ανίχνευση ανωμαλιών π.χ. μια εικόνα που έχει υποστεί photoshop κ.α.

Η εκπαίδευση του αυτόματου κωδικοποιητή, πραγματοποιείται από τον εαυτό του, χωρίς να του παρέχονται προκαθορισμένες ετικέτες, λαμβάνοντας ως είσοδο νέα ακατέργαστα δεδομένα. Επομένως, χρησιμοποιούνται στην μηχανική μάθηση χωρίς επίβλεψη.[10]

Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Python

Εισαγωγή

Η Python είναι μια δημοφιλής γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως σε πολλούς τομείς, συμπεριλαμβανομένης της μηχανικής μάθησης. Επιλέχθηκε ως γλώσσα υλοποίησης της συγκεκριμένης εφαρμογής, για τους εξής λόγους:

- Εύκολη σύνταξη: Η Python έχει μια απλή και καθαρή σύνταξη, που καθιστά εύκολη την κατανόηση και την ανάπτυξη κώδικα. Αυτό καθιστά την Python ιδανική για αρχάριους, αλλά και για έμπειρους προγραμματιστές.
- Εκτεταμένη κοινότητα και οικοσύστημα: Η Python έχει μια τεράστια και ενεργή κοινότητα, που παρέχει πληθώρα βιβλιοθηκών και πλαίσια για μηχανική μάθηση, όπως τα TensorFlow, PyTorch, scikit-learn, pantas, numpy κ.ά.
- Ευελιξία: Η Python είναι μια ευέλικτη γλώσσα, που μπορεί να χρησιμοποιηθεί σε διάφορες μεθόδους μηχανικής μάθησης, συμπεριλαμβανομένης της μάθησης με επίβλεψη, της μάθησης χωρίς επίβλεψη αλλά και ενισχυτικής μάθησης.
- Οικονομική: Η Python είναι δωρεάν και ανοικτού κώδικα, οπότε δεν απαιτείται κόστος για τη χρήση της.

Όπως όλες οι γλώσσες προγραμματισμού, εκτός από πλεονεκτήματα, έχει και κάποια μειονεκτήματα σε σχέση με την εφαρμογή της στην μηχανική μάθηση, που όμως δεν θεωρούνται τόσο σημαντικές για την εφαρμογή μας:

- Απόδοση: Σε ορισμένες περιπτώσεις, η Python μπορεί να μην είναι τόσο αποδοτική όσο άλλες γλώσσες προγραμματισμού, όπως η C++ ή η Java, ειδικά όταν πρόκειται για εφαρμογές, που απαιτούν υψηλή χρονική απόκριση ή επεξεργασία μεγάλων όγκων δεδομένων.

- Περιορισμένες επιδόσεις σε GPU: Ενώ ορισμένες βιβλιοθήκες Python, όπως το TensorFlow και το PyTorch υποστηρίζουν επιτάχυνση μέσω GPU, η Python δεν είναι πάντα η καλύτερη επιλογή, για εφαρμογές που απαιτούν υψηλές επιδόσεις σε GPU.

Παρόλα αυτά, η Python παραμένει μια από τις κορυφαίες επιλογές για την ανάπτυξη μοντέλων μηχανικής μάθησης, λόγω της ευκολίας χρήσης, της εκτεταμένης υποστήριξης και διάθεση πολλαπλών βιβλιοθηκών, καθώς και της ευελιξίας της.

Βιβλιοθήκες Python

Για την υλοποίηση της εφαρμογής μας, επιλέχθηκαν οι παρακάτω βιβλιοθήκες της, οι οποίες βοήθησαν στην απόδοση των λειτουργιών που θεωρήθηκαν απαραίτητες.

pathlip

Είναι μια βιβλιοθήκη που παρέχεται με την Python και χρησιμοποιείται για τη διαχείριση διαδρομών αρχείων και καταλόγων, στο σύστημα αρχείων του υπολογιστή. Η χρήση της, δίνει την δυνατότητα εκτέλεσης πολλών λειτουργιών, όπως η διαχείριση αρχείων και καταλόγων, η δημιουργία, η διαγραφή, η μετονομασία και η ανάγνωση πληροφοριών για αρχεία και καταλόγους.[12]

os

Είναι μια βιβλιοθήκη που παρέχεται με την Python, ιδιαίτερα χρήσιμη για τη διαχείριση αρχείων και την αλληλεπίδραση με το λειτουργικό σύστημα, μέσα από το περιβάλλον της Python.[13]

zipfile

Είναι μια βιβλιοθήκη που παρέχεται με την Python και παρέχει λειτουργίες για την ανάγνωση, την εγγραφή και τη διαχείριση αρχείων ZIP. Δίνεται η δυνατότητα δημιουργία νέων αρχείων ZIP, εξαγωγής από υπάρχοντα αρχεία ZIP και εκτέλεσης άλλων λειτουργιών σχετικών με αρχεία ZIP μέσω της Python.[14]

datetime

Η βιβλιοθήκη datetime της Python, είναι μια ενσωματωμένη βιβλιοθήκη, που παρέχει λειτουργίες για εργασίες που αφορούν ημερομηνίες και χρόνο (time). Επιτρέπει την δημιουργία, ανάλυση και διαχείρισης ημερομηνιών και χρόνων (time) στην Python.[15]

pandas

Παρέχει ισχυρές λειτουργίες για τη φόρτωση, την επεξεργασία, την ανάλυση και την απεικόνιση δεδομένων. Χρησιμοποιεί δομές δεδομένων, οι οποίες είναι οι σειρές (Series) και οι πίνακες (DataFrames), οι οποίοι επιτρέπουν την ευέλικτη αποθήκευση και επεξεργασία δεδομένων. Παρέχει πληθώρα λειτουργιών για την εκτέλεση αναλύσεων δεδομένων, όπως ταξινόμηση, φίλτρα, ομαδοποίηση, συγχώνευση, καθώς και στατιστικές ή μαθηματικές λειτουργίες. Υποστηρίζει διάφορες μεθόδους για τη φόρτωση και την εξαγωγή δεδομένων όπως αρχεία CSV, Excel κ.α. Έχει ενσωματωμένες δυνατότητες, για την απεικόνιση δεδομένων μέσω γραφημάτων και διαγραμμάτων χρησιμοποιώντας την matplotlib και άλλες βιβλιοθήκες.[16]

numpy

Βιβλιοθήκη της Python, που παρέχει υψηλής απόδοσης προγραμματιστικές δομές δεδομένων και εργαλεία για εργασίες με πίνακες πολλαπλών διαστάσεων, καθώς και για εκτέλεση μαθηματικών, λογικών, στατιστικών και άλλων λειτουργιών σε αυτούς τους πίνακες. Χρησιμεύει, όπου απαιτείται ανάλυση δεδομένων και επιστημονικοί υπολογισμοί στην Python. Περιλαμβάνει διάφορες υλοποιήσεις αλγορίθμων, όπως Fast Fourier Transform (FFT), γραμμική άλγεβρα, ταξινόμηση, φιλτράρισμα κ.α.[17]

scikit-learn

Η βιβλιοθήκη scikit-learn, είναι μια από τις πιο δημοφιλείς βιβλιοθήκες για την μηχανική μάθηση στη γλώσσα προγραμματισμού Python. Παρέχει ένα εύχρηστο περιβάλλον για την ανάπτυξη μοντέλων μηχανικής μάθησης, συμπεριλαμβανομένων αλγορίθμων για την κατηγοριοποίηση (classification), την παρεμβολή (regression), την συσταδοποίηση (clustering), την ανάλυση δεδομένων και την προ-επεξεργασία (preproccesing). Μερικά από τα βασικά χαρακτηριστικά και λειτουργίες που εμπεριέχονται στην βιβλιοθήκη scikit-learn είναι τα εξής:

- **Μάθηση με επίβλεψη (Supervised Learning):** Παρέχει αλγορίθμους για προβλήματα κατηγοριοποίησης και παρεμβολής.
- **Μάθηση χωρίς επίβλεψη (Unsupervised Learning):** Περιλαμβάνει αλγορίθμους για τη συσταδοποίηση, τη μείωση της διάστασης και την ανίχνευση ανωμαλιών, χωρίς τη χρήση ετικετών.
- **Εκτίμηση της απόδοσης μοντέλου (Model Evaluation):** Παρέχει εργαλεία για την εκτίμηση και τη βελτιστοποίηση της απόδοσης των μοντέλων, συμπεριλαμβανομένων διαδικασιών σταθεροποίησης (όπως η διασταυρούμενη επικύρωση).
- **Προ-επεξεργασία δεδομένων (Data Preprocessing):** Παρέχει εργαλεία για την επεξεργασία και τον καθαρισμό των δεδομένων πριν την εκπαίδευση του μοντέλου, όπως κλιμακοποίηση (Feature Scaling), κωδικοποίηση μεταβλητών κατηγοριών, κανονικοποίηση (normalization), ανίχνευση και εξάλειψη ακραίων σημείων (Outlier Detection and Removal) και αντιμετώπιση κενών τιμών.
- **Επιλογή χαρακτηριστικών (Feature Selection):** Παρέχει αλγορίθμους για την αυτόματη επιλογή των πιο σημαντικών χαρακτηριστικών που συμβάλλουν στην απόδοση του μοντέλου.
- **Αντιμετώπιση ανισορροπίας κατηγοριών (Imbalanced Class Handling):** Παρέχει τεχνικές για την αντιμετώπιση του προβλήματος της ανισορροπίας κλάσεων σε προβλήματα κατηγοριοποίησης.

Η βιβλιοθήκη scikit-learn είναι εύκολη στη χρήση, καλά τεκμηριωμένη και παρέχει πλούσια λειτουργικότητα, για την ανάπτυξη και αξιολόγηση μοντέλων μηχανικής μάθησης. Είναι ιδιαίτερα χρήσιμη, σε έργα μηχανικής μάθησης και ανάλυσης δεδομένων.[18]

sklearn.preprocessing.StandardScaler

Η `StandardScaler` ανήκει στη βιβλιοθήκη `sklearn.preprocessing` της scikit-learn και χρησιμοποιείται για την προ-επεξεργασία δεδομένων, πριν την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Συγκεκριμένα, η `StandardScaler` εκτελεί την κλιμάκωση (scaling) των χαρακτηριστικών (features) ενός dataset, με τη μέθοδο του Z-score scaling.
Ο Z-score scaling, γνωστός και ως standardization, μετατρέπει κάθε χαρακτηριστικό του dataset έτσι ώστε να έχει μέση τιμή 0 και τυπική απόκλιση 1. Αυτό επιτυγχάνεται αφαιρώντας τη μέση τιμή του χαρακτηριστικού και διαιρώντας με την τυπική απόκλιση. Η διαδικασία αυτή εξισορροπεί τα δεδομένα, κάνοντάς τα πιο κατανοητά για τους αλγορίθμους μηχανικής μάθησης, καθώς απομακρύνει την επίδραση των μεγεθών των χαρακτηριστικών στην εκπαίδευση του μοντέλου.
Η χρήση της `StandardScaler`, συνήθως γίνεται ως μέρος μιας προ-επεξεργασίας δεδομένων, πριν την εκπαίδευση ενός μοντέλου μηχανικής μάθησης, προκειμένου να βελτιστοποιηθεί η απόδοση του μοντέλου.

sklearn.preprocessing.RobustScaler

Ο `RobustScaler` ανήκει στη βιβλιοθήκη `sklearn.preprocessing` της scikit-learn και χρησιμοποιείται, για την προ-επεξεργασία δεδομένων πριν την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Αντίθετα από τον `StandardScaler`, ο `RobustScaler` χρησιμοποιεί μετασχηματισμούς, που είναι ανθεκτικοί σε ακραίες τιμές (outliers) στα δεδομένα.
Οι ακραίες τιμές μπορούν να επηρεάσουν αρνητικά την απόδοση ενός μοντέλου μηχανικής μάθησης, ειδικά όταν χρησιμοποιούνται μέθοδοι που είναι ευαίσθητες σε

αλλαγές στην κλίμακα των δεδομένων. Ο `RobustScaler` είναι σχεδιασμένος για να αντιμετωπίζει αυτό το πρόβλημα.

Η διαδικασία που ακολουθεί ο `RobustScaler` είναι η εξής:

- Υπολογίζει την μέση τιμή και τη διακύμανση κάθε χαρακτηριστικού.
- Αντί να χρησιμοποιήσει απλή μέση τιμή και διακύμανση, χρησιμοποιεί τη μέση τιμή (median) και το διατεταρτημοριακό εύρος (interquartile range, IQR) των χαρακτηριστικών. Αυτό το καθιστά ανθεκτικό στις ακραίες τιμές.
- Κατόπιν, κάνει την κλιμάκωση των χαρακτηριστικών αναλόγως της μέσης τιμής και του διατεταρτημοριακού εύρους.

Η `RobustScaler` είναι χρήσιμη, όταν τα δεδομένα περιέχουν ακραίες τιμές ή δεν ακολουθούν κανονική κατανομή, καθώς εξασφαλίζει ότι η κλιμάκωση των δεδομένων, θα είναι ανθεκτική σε αυτές τις αποκλίσεις.

sklearn.preprocessing.MinMaxScaler

Η `MinMaxScaler` ανήκει επίσης στη βιβλιοθήκη `sklearn.preprocessing` της scikit-learn και χρησιμοποιείται για την προ-επεξεργασία δεδομένων, πριν την εκπαίδευση ενός μοντέλου μηχανικής μάθησης.

Ο `MinMaxScaler` εκτελεί την κλιμάκωση (scaling) των χαρακτηριστικών (features) ενός dataset, σε ένα εύρος προκαθορισμένων τιμών διαστήματος, συνήθως το διάστημα [0, 1]. Αυτό επιτυγχάνεται με τον ακόλουθο τρόπο:

- Υπολογίζει το εύρος (range) κάθε χαρακτηριστικού, δηλαδή το μέγιστο και το ελάχιστο τους.
- Κάνει την κλιμάκωση των χαρακτηριστικών, χρησιμοποιώντας την παρακάτω μετασχηματισμένη τιμή για κάθε χαρακτηριστικό:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

όπου X, είναι η αρχική τιμή του χαρακτηριστικού, X_{min} είναι το ελάχιστο εύρος του χαρακτηριστικού και X_{max} είναι το μέγιστο εύρος του χαρακτηριστικού.

Η `MinMaxScaler` είναι χρήσιμη, όταν οι προβλέψεις που προκύπτουν από το μοντέλο πρέπει να ανακτήσουν τις αρχικές κλίμακες των δεδομένων, είτε για λόγους ερμηνείας, είτε για ενσωμάτωση σε συστήματα όπου η ανακατασκευή της πρωτοτυπίας των δεδομένων είναι σημαντική.
Επιπλέον, μια συνήθης πρακτική είναι να χρησιμοποιηθεί η `MinMaxScaler` πριν από αλγόριθμους που βασίζονται σε αποστάσεις, όπως οι k-Nearest Neighbors (k-NN) ή οι Support Vector Machines (SVMs), για να βελτιστοποιηθεί η απόδοσή τους.

sklearn.preprocessing.normalize

Η 'normalize' ανήκει στη βιβλιοθήκη `sklearn.preprocessing` της scikit-learn και χρησιμοποιείται για την κανονικοποίηση (normalization) των δεδομένων, σε ένα επιλεγμένο κανονικοποιημένο εύρος. Η κανονικοποίηση, είναι μια διαδικασία μετασχηματισμού των δεδομένων, ώστε να έχουν μοναδιαία νόρμα (μήκος 1) ή να είναι στο εύρος [0, 1] ή [-1, 1], ανάλογα με την επιλογή.
Η διαδικασία της κανονικοποίησης, συνήθως εφαρμόζεται σε κάθε σειρά (γραμμή) των δεδομένων ξεχωριστά. Ανάλογα με την επιλεγμένη νόρμα, μπορεί να χρησιμοποιηθούν διαφορετικές μέθοδοι για τον υπολογισμό της.

- Η L1 νόρμα, γνωστή και ως "Manhattan distance" ή "Taxicab norm", υπολογίζει το άθροισμα των απόλυτων τιμών των δεδομένων.
- Η L2 νόρμα, γνωστή και ως "Euclidean distance", υπολογίζει την Ευκλείδεια απόσταση των δεδομένων.
- Η Max νόρμα, παίρνει το μέγιστο απόλυτο στοιχείο του κάθε δείγματος.

Η κανονικοποίηση δεδομένων, μπορεί να είναι χρήσιμη σε αλγόριθμους μηχανικής μάθησης, που είναι ευαίσθητοι στην κλίμακα των δεδομένων ή όταν θέλουμε να φέρουμε τις τιμές των δεδομένων, σε μια κοινή κλίμακα, για να μπορούν να συγκριθούν εύκολα. Επίσης, μπορεί να βοηθήσει στην επιτάχυνση της σύγκλισης των αλγορίθμων μηχανικής μάθησης.

sklearn.decomposition.PCA

Η ‘PCA’ ανήκει στην βιβλιοθήκη `sklearn.decomposition` της scikit-learn και χρησιμοποιείται, για την εκτέλεση της Ανάλυσης Κύριων Συνιστωσών (Principal Component Analysis - PCA). Η PCA, είναι μια τεχνική μείωσης της διάστασης των δεδομένων, που χρησιμοποιείται συχνά στην ανάλυση και την εξόρυξη δεδομένων.
Η κύρια ιδέα της PCA, είναι να μετατρέψει ένα σύνολο δεδομένων που μπορεί να έχει πολλές διαστάσεις, σε ένα νέο σύνολο δεδομένων που περιέχει λιγότερες διαστάσεις, διατηρώντας παράλληλα όσον το δυνατόν την περισσότερη πληροφορία.
Η PCA λειτουργεί με τα ακόλουθα βασικά βήματα:

- Κεντράρισμα των δεδομένων: Αφαιρεί τη μέση τιμή από κάθε χαρακτηριστικό, ώστε να έχουν μέση τιμή 0.
- Υπολογισμός των Κύριων Συνιστωσών (Principal Components): Υπολογίζει μια νέα σειρά από γραμμικές συνιστώσες, που αντιπροσωπεύουν τη μέγιστη δυνατή διακύμανση των δεδομένων.
- Επιλογή Διαστάσεων: Επιλέγει τις κύριες συνιστώσες με τη μεγαλύτερη διακύμανση, δηλαδή εκείνες που διατηρούν το μεγαλύτερο μέρος της πληροφορίας.
- Μείωση της Διάστασης: Τέλος, εφαρμόζει τη μετατροπή στα δεδομένα που δίνονται ώστε να απεικονίζονται στις νέες κύριες συνιστώσες, ενώ μπορεί να μειώσει τις διαστάσεις των δεδομένων.

Η PCA είναι χρήσιμη για μείωση της διάστασης των δεδομένων, προτού εφαρμόσουμε έναν αλγόριθμο μηχανικής μάθησης, καθώς μπορεί να βοηθήσει στην αποφυγή της υπέρ-εκπαίδευσης και να εξασφαλίσει την αποτελεσματικότητα του μοντέλου.

sklearn.metrics.silhouette_score

Η ‘silhouette_score’ ανήκει στην βιβλιοθήκη `sklearn.metrics` της scikit-learn και χρησιμοποιείται ως δείκτης εσωτερικής μετρικής αξιολόγησης, για να αξιολογήσει την ποιότητα της συσταδοποίησης που παράγεται. Αυτή η μετρική, παρέχει έναν αριθμητικό δείκτη, που μετρά το πόσο καλά ένα σημείο ταιριάζει στη συστάδα του, σε σχέση με τις άλλες συστάδες. Αυτό είναι χρήσιμο στην αξιολόγηση της συνέπειας

της συσταδοποίησης και στον εντοπισμό σημείων δεδομένων με κακή συσταδοποίηση.

Ο δείκτης κυμαίνεται μεταξύ -1 και 1, όπου μεγαλύτερες τιμές υποδεικνύουν καλύτερη συσταδοποίηση. Η καλύτερη τιμή είναι 1 και η χειρότερη τιμή είναι -1. Οι τιμές κοντά στο 0 υποδεικνύουν επικαλυπτόμενες συστάδες. Οι αρνητικές τιμές, γενικά υποδεικνύουν ότι ένα δείγμα έχει εκχωρηθεί σε λάθος συστάδα, καθώς μια διαφορετική συστάδα είναι πιο παρόμοια.

sklearn.metrics.calinski_harabasz_score

Η ‘calinski_harabasz_score’ ανήκει στην βιβλιοθήκη `sklearn.metrics` της scikit-learn και χρησιμοποιείται για την αξιολόγηση της καλής συσταδοποίησης. Βασίζεται στην αναλογία του μέσου όρου διασποράς μεταξύ συστάδων, προς το μέσο όρο διασποράς εντός μιας συστάδας. Μια υψηλότερη βαθμολογία Calinski-Harabasz, υποδηλώνει καλύτερα καθορισμένες συστάδες.
Στην πράξη, η βαθμολογία Calinski-Harabasz χρησιμοποιείται συχνά για τη σύγκριση διαφορετικών λύσεων συσταδοποίησης, όπου μια υψηλότερη βαθμολογία υποδηλώνει μια καλύτερα καθορισμένη δομή συσταδοποίησης. Ωστόσο, είναι σημαντικό να σημειωθεί ότι έχει ορισμένους περιορισμούς, όπως η ευαισθησία στην κλίμακα των δεδομένων και το σχήμα των συστάδων. Επομένως, θα πρέπει να χρησιμοποιείται σε συνδυασμό με άλλες μετρήσεις αξιολόγησης και γνώσεις τομέα κατά την αξιολόγηση των αποτελεσμάτων ομαδοποίησης.

sklearn.metrics.davies_bouldin_score

Η ‘davies_bouldin_score’ ανήκει στην βιβλιοθήκη `sklearn.metrics` της scikit-learn και χρησιμοποιείται για την αξιολόγηση της ποιότητας των αλγορίθμων συσταδοποίησης. Προσδιορίζει ποσοτικά το πόσο συμπαγές είναι οι συστάδες, καθώς και το πόσο καλό διαχωρισμό έχουν. Ένας χαμηλότερος δείκτης Davies-Bouldin υποδηλώνει καλύτερη ομαδοποίηση. Υπολογίζεται ως η μέση ομοιότητα μεταξύ κάθε συστάδας και της πιο παρόμοιας συστάδας, όπου η ομοιότητα, ορίζεται ως ο λόγος της διασποράς εντός συστάδας, προς τον διαχωρισμό μεταξύ συστάδων.

Ένας χαμηλότερος δείκτης Davies-Bouldin υποδηλώνει καλύτερη συσταδοποίηση, καθώς υποδηλώνει ότι οι συστάδες είναι πιο συμπαγές και καλά διαχωρισμένες. Όπως και άλλες μετρήσεις αξιολόγησης συσταδοποίησης, θα πρέπει να χρησιμοποιείται σε συνδυασμό με άλλα μέτρα και γνώσεις τομέα, για την αποτελεσματική αξιολόγηση της απόδοσης των αλγορίθμων συσταδοποίησης.

sklearn.cluster.KMeans

Η 'KMeans' ανήκει στην βιβλιοθήκη `sklearn.cluster` της scikit-learn και χρησιμοποιείται για την εκτέλεση του αλγορίθμου ομαδοποίησης K-Means. Ο αλγόριθμος K-Means, είναι ένας από τους πιο δημοφιλείς αλγορίθμους ομαδοποίησης και χρησιμοποιείται για τον διαχωρισμό ενός συνόλου δεδομένων, σε ομάδες με βάση την ομοιότητα μεταξύ των δειγμάτων. Παρέχει υλοποίηση του αλγορίθμου συσταδοποίησης (clustering) K-Means, στο πλαίσιο της scikit-learn. Ο αλγόριθμος K-Means είναι ένας από τους πιο δημοφιλείς και απλούς αλγορίθμους συσταδοποίησης.
Οι βασικές έννοιες και λειτουργίες του αλγορίθμου K-Means είναι οι εξής:

- Αρχικοποίηση: Επιλογή τυχαίων κέντρων για τις συστάδες. Ο αριθμός των συστάδων είναι προκαθορισμένος.
- Ανάθεση: Κάθε δείγμα ανατίθεται στη συστάδα με το πλησιέστερο κέντρο, με βάση ένα ορισμένο μετρικό απόστασης, συνήθως την ευκλείδεια απόσταση.
- Ανανέωση Κέντρων: Τα κέντρα των συστάδων ανανεώνονται ως η μέση τιμή των δεδομένων που ανήκουν σε αυτή τη συστάδα.
- Επανάληψη: Οι δύο προηγούμενες διαδικασίες επαναλαμβάνονται μέχρι να συγκλίνει ο αλγόριθμος, δηλαδή όταν τα κέντρα των συστάδων δεν αλλάζουν πολύ μεταξύ δύο συνεχόμενων επαναλήψεων.

Ο K-Means είναι ιδιαίτερα αποτελεσματικός για σύνολα δεδομένων με μεγάλο αριθμό δειγμάτων και όταν οι συστάδες είναι περίπου σφαιρικές και έχουν παρόμοια μεγέθη. Ωστόσο, είναι ευάλωτος σε ακραίες τιμές (outliers) και δεν είναι πάντα αποτελεσματικός για δεδομένα μη γραμμικά διαχωρίσιμα. Παρόλα αυτά, αποτελεί έναν από τους πρώτους αλγορίθμους, που εξετάζονται για προβλήματα συσταδοποίησης, λόγω της απλότητάς του και της ευκολίας χρήσης του.

scipy.stats.mstats.winsorize

Η winsorize, είναι συνάρτηση που παρέχεται από τη βιβλιοθήκη scipy.stats της Python.
Χρησιμοποιείται για την εφαρμογή της μεθόδου Winsorization, σε ένα σύνολο δεδομένων. Η Winsorization, είναι μια τεχνική για την εξομάλυνση ακραίων τιμών (outliers), σε ένα σύνολο δεδομένων. Κατά τη χρήση της μεθόδου Winsorization, οι ακραίες τιμές στις ακραίες πλευρές της κατανομής, αντικαθίστανται από τις τιμές των κοντινότερων παρατηρήσεων, που δεν είναι ακραίες. Αυτό μπορεί να γίνει για ένα ή περισσότερα ποσοστά των ακραίων τιμών, στην κάθε άκρη της κατανομής. Είναι χρήσιμη, για την εξομάλυνση ακραίων τιμών σε δεδομένα, πριν εκτελεστούν στατιστικές αναλύσεις ή μοντέλα μηχανικής μάθησης, καθώς μπορεί να βελτιώσει την ακρίβεια και την απόδοση του μοντέλου σας, αφαιρώντας την επίδραση των ακραίων τιμών.[19]

seaborn

Το Seaborn είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων Python, που βασίζεται στη βιβλιοθήκη matplotlib. Παρέχει μια διεπαφή υψηλού επιπέδου, για τη σχεδίαση ελκυστικών και ενημερωτικών στατιστικών γραφικών.[20]

dataframe_image

Βιβλιοθήκη για τη μετατροπή pandas DataFrames ως εικόνες. Πραγματοποιεί, απευθείας εξαγωγή των δεδομένων, από το DataFrame ως εικόνα τύπου png.[21]

matplotlib.pyplot

Η βιβλιοθήκη matplotlib.pyplot είναι μέρος της δημοφιλούς βιβλιοθήκης Python για τη δημιουργία γραφημάτων, η οποία ονομάζεται Matplotlib. Το pyplot είναι ένα υποπακέτο της Matplotlib, το οποίο παρέχει μια παρόμοια λειτουργικότητα με αυτή της βιβλιοθήκης MATLAB. Επιτρέπει στους χρήστες να δημιουργήσουν διάφορα είδη γραφημάτων, όπως γραφήματα γραμμών, διασποράς, ιστογράμματα, πίτες, κ.λπ. Είναι μια ισχυρή βιβλιοθήκη, που χρησιμοποιείται ευρέως στον τομέα της ανάλυσης δεδομένων, της επιστημονικής ανάλυσης και άλλων πεδίων για τη δημιουργία εντυπωσιακών γραφημάτων και αναπαραστάσεων δεδομένων. Προσφέρει μια πιο απλή διεπαφή, σε σύγκριση με το βασικό API της Matplotlib, καθιστώντας ευκολότερη την αρχικοποίηση των γραφημάτων και την εκτύπωση τους.[22]

ΠΡΟΒΛΕΨΗ ΟΔΗΓΙΚΗΣ ΣΥΜΠΕΡΙΦΟΡΑΣ

Εισαγωγή

Η οδήγηση είναι μέρος της καθημερινότητάς για πολλούς ανθρώπους. Λόγω της συχνότητας που πραγματοποιείται αυτή η λειτουργία, θεωρείται εσφαλμένα μια απλή λειτουργία, ενώ στην πραγματικότητα είναι ένα από τα πιο δύσκολα καθήκοντα, γιατί ο οδηγός είναι υπεύθυνος για μια σειρά από ευθύνες επιπρόσθετες της οδήγησης. Η πρωταρχική ευθύνη ενός οδηγού βέβαια πρέπει να είναι ο σωστός χειρισμός του οχήματος, με αποκλειστική συγκέντρωση στην οδήγηση. Ταυτόχρονα όμως επωμίζεται και με διάφορες δευτερεύουσες δουλειές.

Η δυνατότητα μοντελοποίησης της ρεαλιστικής οδηγικής συμπεριφοράς είναι χρήσιμη με εφαρμογές ποιότητας, που είναι επωφελείς για τη βελτίωση της ασφάλειας του οδηγού, την αποφυγή τροχαίων ατυχημάτων, τη μείωση της κατανάλωσης καυσίμου και στις αυτοκινητοβιομηχανίες στο σύνολό τους.

Ένας από τους ικανούς βοηθούς για την υλοποίηση τέτοιας μοντελοποίησης, είναι τα έξυπνα κινητά τηλέφωνα (smartphones), τα οποία περιλαμβάνουν μια ποικιλία ενσωματωμένων αισθητήρων, όπως επιταχυνσιόμετρα, γυροσκόπια, GPS και κάμερες, τα οποία μπορούν να συνεισφέρουν σε δεδομένα, χρήσιμα για την ανάλυση των μοτίβων συμπεριφοράς του οδηγού και τη συλλογή δεδομένων του οχήματος. Η γραμμική επιτάχυνση της κίνησης μετριέται με επιταχυνσιόμετρο, ενώ η γωνιακή ταχύτητα περιστροφής μετριέται με γυροσκόπιο. Και οι δύο αισθητήρες μετρούν το ρυθμό αλλαγής, με διαφορά ότι μετρούν διαφορετικά πράγματα.

Ανάλυση και Σχεδιασμός Εφαρμογής

Ο σκοπός της εφαρμογής είναι η ανάλυση και πρόγνωση της οδηγικής συμπεριφοράς, βασιζόμενη σε δεδομένα από επτά διαφορετικούς οδηγούς, καθένας και διαφορετική ημέρα, οι οποίοι πραγματοποιούν την ίδια διαδρομή και προς τις δύο κατευθύνσεις, δηλαδή από την αφετηρία στον προορισμό και από τον προορισμό στην αφετηρία. Οι διαδρομές πραγματοποιούνται σε διαφορετικές χρονικές στιγμές της ημέρας (πρωί, μεσημέρι, απόγευμα, βράδυ).

Η εφαρμογή καλείται να αναλύσει τα δεδομένα, αναγνωρίζοντας μοτίβα και να εξάγει συμπεράσματα ως προς την οδηγική συμπεριφορά, κατατάσσοντας τα σε κατηγορίες, οι οποίες δεν είναι προκαθορισμένες, ούτε ως προς την ονομασία και χαρακτηρισμό τους, ούτε προς την ποσότητά τους. Ο σκοπός είναι δηλαδή, να υπάρξει ανίχνευση μοτίβων, έτσι ώστε να μπορούν να εξαχθούν ασφαλή συμπεράσματα, εάν η οδηγική συμπεριφορά ενός οδηγού είναι φυσιολογική, επιθετική ή επικίνδυνη ή ακόμα και πιο αναλυτικά, όπως εάν είναι επιθετική ή επικίνδυνη, ως προς την επιτάχυνση ή την επιβράδυνση της ταχύτητας ή ακόμα και προς τους απότομους ελιγμούς και στροφές.

Τα δεδομένα θα υποστούν όλες τις απαραίτητες λειτουργίες προ-επεξεργασίας και μετασχηματισμού, έτσι ώστε να καταλήξουν σε μορφή κατάλληλη για την εφαρμογή μας. Αυτές οι λειτουργίες ενσωματώνουν διαδικασίες, όπως ενσωμάτωση σε ένα σύνολο όλων των δεδομένων από τα πολλαπλά αρχεία, αντικατάσταση τυχόν κενών τιμών, απαλοιφή διπλότυπων εγγραφών, μετονομασία χαρακτηριστικών, δημιουργία νέων χαρακτηριστικών βασιζόμενων στα ήδη υπάρχοντα, μετασχηματισμός των τιμών ήδη υπαρχόντων χαρακτηριστικών, καθώς και διαδικασίες κλιμάκωσης και κανονικοποίησης χαρακτηριστικών και διαδικασίας μείωσης διαστάσεων των χαρακτηριστικών.

Από την φύση των δεδομένων, καθώς και τον σκοπό της εφαρμογής μας, έχουμε καταλήξει στο συμπέρασμα ότι θα γίνει χρήση αλγορίθμων μηχανικής μάθησης χωρίς επίβλεψη, μιας και δεν έχουμε στην διάθεσή μας προκαθορισμένες κατηγορίες, ούτε σύνολο δεδομένων για εκπαίδευση του μοντέλου μας. Η προσέγγιση των δεδομένων θα γίνει αρχικά ως προς το σύνολό τους, με σκοπό να ανιχνευτούν μοτίβα στο σύνολο των οδηγικών συμπεριφορών. Αυτή η διαδικασία, θα επιτευχθεί με εκτέλεση αλγόριθμου μηχανικής μάθησης χωρίς επίβλεψη.

Ο αλγόριθμος που επιλέχθηκε είναι ο **K-means**. Είναι ένας επαναληπτικός αλγόριθμος, που προσπαθεί να χωρίσει το σύνολο δεδομένων σε προκαθορισμένες από το K διακριτές μη επικαλυπτόμενες υποομάδες (συστάδες), όπου κάθε σημείο δεδομένων ανήκει σε μία μόνο συστάδα. Προσπαθεί να κάνει τα σημεία δεδομένων εντός της συστάδας, όσο το δυνατόν πιο όμοια, ενώ παράλληλα διατηρεί τις συστάδες

όσο το δυνατόν πιο διαφορετικά (μακριά). Εκχωρεί σημεία δεδομένων σε μία συστάδα, έτσι ώστε το άθροισμα της τετραγωνικής απόστασης μεταξύ των σημείων δεδομένων και του κέντρου της συστάδας (αριθμητικός μέσος όρος όλων των σημείων δεδομένων που ανήκουν σε αυτή τη συστάδα), να είναι στο ελάχιστο. Όσο λιγότερη διακύμανση έχουμε μέσα σε συστάδες, τόσο πιο ομοιογενή (παρόμοια) τα σημεία δεδομένων είναι μέσα στην ίδια συστάδα. Είναι απλός και κατανοητός αλγόριθμος στην εφαρμογή του. Μπορεί να χειριστεί αποτελεσματικά μεγάλα σύνολα δεδομένων, καθιστώντας το κατάλληλο για εφαρμογές με μεγάλο όγκο δεδομένων. Τα αποτελέσματα συσταδοποίησης που παράγονται από το K-means, είναι σχετικά εύκολο να ερμηνευτούν, καθώς κάθε συστάδα αντιπροσωπεύεται από το κέντρο του. Αυτό επιτρέπει στους χρήστες, να κατανοούν τα χαρακτηριστικά διαφορετικών συμπλεγμάτων και να ερμηνεύουν τα αποτελέσματα στο πλαίσιο του τομέα εφαρμογής, όπως στην περίπτωσή μας ο εντοπισμός διακριτών οδηγικών συμπεριφορών, με βάση τα δεδομένα αισθητήρων. Επίσης είναι υπολογιστικά αποδοτικό, ειδικά για μεγάλα σύνολα δεδομένων, καθώς έχει χρονική πολυπλοκότητα O(n*k*d), όπου n είναι ο αριθμός των σημείων δεδομένων, k είναι ο αριθμός των συστάδων και d είναι ο αριθμός των διαστάσεων. Στα ελάττώματά του, είναι ότι προϋποθέτει τον ορισμό του αριθμού των συστάδων (K) που θα αποτυπώσει. Επίσης, η λειτουργία του να μεγιστοποιεί τη συνοχή και να ελαχιστοποιεί τον διαχωρισμό, οδηγεί στον σχηματισμό ασύνδετων συστάδων, με αποτέλεσμα να μην μπορεί να διαχειριστεί φωλιασμένες συστάδες. Λόγω αυτής της λειτουργίας, έχει την τάση δημιουργίας σφαιρικών συστάδων, καθώς και τον καθιστά ευαίσθητο στην παρουσία ακραίων σημείων.

Για την επίλυση αυτού του προβλήματος εύρεσης βέλτιστου αριθμού συστάδων (K), θα επιστρατευτούν τεχνικές και μέθοδοι κατάλληλοι για την εύρεση του βέλτιστου αριθμού συστάδων, που θα διοχετευτεί στον αλγόριθμο k-means. Οι μέθοδοι αυτοί μετρούν την ποιότητα της συσταδοποίησης, βασιζόμενοι στην ιδέα ότι η καλή συσταδοποίηση, έχει μικρή εσωτερική απόσταση μεταξύ των στοιχείων της ίδιας συστάδας και μεγάλη απόσταση μεταξύ των κέντρων των διαφορετικών συστάδων. Αυτές οι μέθοδοι είναι οι εξής:

1. *Μέθοδος Elbow:* Χρησιμοποιεί την αδράνεια (inertia), για την μέτρηση της αποτελεσματικότητας ενός συγκεκριμένου αριθμού συστάδων. Όσο

χαμηλότερη η αδράνεια, τόσο πιο σφιχτά συγκεντρώνονται τα σημεία δεδομένων γύρω από τα κεντροειδή τους.

2. *Silhouette Score:* Μετρά τον διαχωρισμό μεταξύ των συστάδων και τη συνοχή εντός των συστάδων. Ένα υψηλό αποτέλεσμα, υποδεικνύει καλά διαχωρισμένες συστάδες με ελάχιστη επικάλυψη.
3. *Calinski-Harabasz Index:* Αξιολογεί την αναλογία της διασποράς μεταξύ των συστάδων, προς τη διασπορά εντός της συστάδας. Υψηλός δείκτης υποδεικνύει υψηλό διαχωρισμό μεταξύ των συστάδων, σε σχέση με το πόσο συμπαγές είναι οι συστάδες.
4. *Davies-Bouldin Index:* Μετρά τη μέση ομοιότητα μεταξύ κάθε συστάδας και της πιο παρόμοιας συστάδας. Χαμηλός δείκτης υποδεικνύει καλά διαχωρισμένες συστάδες, με ελάχιστη διακύμανση εντός της συστάδας.

Οι παραπάνω μέθοδοι θα οπτικοποιηθούν σε διάγραμμα αποτύπωσης της μέτρησής τους, σε σχέση με ένα εύρος υποψήφιων αριθμών συστάδων (2-10). Με αυτό τον τρόπο εκεί που θα γίνει σύγκλιση των μεθόδων, θα γίνει η επιλογή του βέλτιστου αριθμού συστάδων από τον χρήστη.

Επίσης, θα εφαρμοστούν διαφορετικοί μέθοδοι ανίχνευσης ακραίων σημείων, κλιμάκωσης και κανονικοποίησης χαρακτηριστικών, ακόμα και συνδυασμός δύο ή περισσοτέρων εξ αυτών. Με αυτόν τον τρόπο, θα μπορούν να υπάρξουν αξιόλογα συμπεράσματα μέσω της διαδικασίας της σύγκρισης. Οι μέθοδοι που θα χρησιμοποιηθούν είναι οι εξής:

1. *Ανίχνευση Ακραίων Σημείων (Winsorize):* Είναι μια τεχνική για την εξομάλυνση ακραίων τιμών (outliers) σε ένα σύνολο δεδομένων. Κατά τη χρήση της μεθόδου Winsorization, οι ακραίες τιμές στις ακραίες πλευρές της κατανομής, αντικαθίστανται από τις τιμές των κοντινότερων παρατηρήσεων που δεν είναι ακραίες. Αυτό μπορεί να γίνει για ένα ή περισσότερα ποσοστά των ακραίων τιμών, στην κάθε άκρη της κατανομής. Είναι χρήσιμη για την εξομάλυνση ακραίων τιμών σε δεδομένα, πριν εκτελεστούν μοντέλα μηχανικής μάθησης, καθώς μπορεί να βελτιώσει την ακρίβεια και την απόδοση του μοντέλου, αφαιρώντας την επίδραση των ακραίων τιμών. Λαμβάνοντας υπόψη το γεγονός, ότι ο αλγόριθμος K-means είναι επιρρεπής στα ακραία σημεία και το γεγονός, ότι τα δεδομένα μας απαρτίζονται από

τιμές αισθητήρων επιταχυνσιόμετρου και γυροσκοπίου, δεν επιθυμούμε την εξάλειψη των ακραίων σημείων μιας και μπορεί να υποδεικνύουν ακραίες οδηγικές συμπεριφορές. Επομένως, η χρήση αυτής της τεχνικής, μπορεί να φανεί χρήσιμη για να μας οδηγήσει σε βέλτιστο αποτέλεσμα.

2. *Μέθοδοι κλιμάκωσης χαρακτηριστικών*

   Οι τεχνικές κλιμάκωσης προσαρμόζουν την κλίμακα των δεδομένων, χωρίς να αλλάζουν την κατανομή τους.

   i. *StandardScaler*: Είναι ένας γρήγορος και εξειδικευμένος αλγόριθμος για την κλιμάκωση δεδομένων. Υπολογίζει τη μέση και τυπική απόκλιση του συνόλου δεδομένων και την κανονικοποιεί, αφαιρώντας τη μέση τιμή και διαιρώντας με την τυπική απόκλιση. Η χρήση του StandardScaler είναι χρήσιμη, εάν το σύνολο δεδομένων ακολουθεί μια κανονική κατανομή. Είναι ευαίσθητο σε ακραίες τιμές και τα χαρακτηριστικά ενδέχεται να έχουν διαφορετική κλίμακα μεταξύ τους, παρουσία ακραίων τιμών.

   ii. *MinMaxScaler*: Είναι μια απλή και αποτελεσματική συνάρτηση γραμμικής κλίμακας. Κλιμακώνει το σύνολο δεδομένων μεταξύ 0 και 1. Με άλλα λόγια, οι ελάχιστες και μέγιστες τιμές στο κλιμακωτό σύνολο δεδομένων, είναι 0 και 1 αντίστοιχα. Το MinMaxScaler χρησιμοποιείται συχνά ως εναλλακτική του Standard Scaler, εάν ο μηδενικός μέσος όρος και η μοναδιαία διακύμανση, θέλουν να αποφευχθούν. Συμπιέζει τις τιμές σε πολύ στενό εύρος. Είναι ευαίσθητο σε ακραίες τιμές, ειδικά όταν χρησιμοποιείτε ένα στενό εύρος για κλιμάκωση. Για να μετριαστεί ο αντίκτυπος των ακραίων τιμών, θα οριστεί ένα ευρύτερο εύρος για την κλιμάκωση, το οποίο θα είναι (-1, 1). Αυτό επιτρέπει στη κλιμάκωση να απλώνει περισσότερο τις τιμές και να μειώνει τον αντίκτυπο των ακραίων τιμών. Αυτό το εύρος επίσης είναι πιο σωστό, λαμβάνοντας υπόψη ότι τα δεδομένα περιέχουν αρνητικές τιμές. Το MinMaxScaler, δεν λαμβάνει υπόψη τη μορφή της κατανομής των δεδομένων. Επομένως, αν τα δεδομένα δεν ακολουθούν μια κανονική κατανομή, αυτό δεν θα επηρεάσει απαραιτήτως την απόδοση του.

iii. *RobustScaler*: Είναι μια τεχνική που χρησιμοποιεί διάμεσο και τεταρτημόριο, για να αντιμετωπίσει τις ακραίες τιμές. Αντί να αφαιρέσει τη μέση τιμή, το RobustScaler αφαιρεί τη διάμεση τιμή και κλιμακώνει τα δεδομένα σύμφωνα με το εύρος ποσοστών, γνωστό και ως IQR: Interquartile Range. Από προεπιλεγμένες ρυθμίσεις, το IQR είναι το εύρος μεταξύ του 1ου τεταρτημορίου (25ο τεταρτημόριο) και του 3ου τεταρτημορίου (75ο τεταρτημόριο).
Σε αντίθεση με τους προηγούμενους κλιμακωτές, τα στατιστικά στοιχεία κεντραρίσματος και κλιμάκωσης του RobustScaler βασίζονται σε εκατοστημόρια και επομένως δεν επηρεάζονται από έναν μικρό αριθμό πολύ μεγάλων οριακών ακραίων τιμών. Κατά συνέπεια, το εύρος των τιμών των μετασχηματισμένων χαρακτηριστικών που προκύπτει, είναι μεγαλύτερο από ό,τι για τους προηγούμενους κλιμακωτές και το πιο σημαντικό, είναι περίπου παρόμοιες. Εκτός από τα ακραία σημεία, χειρίζεται και την λοξότητα των δεδομένων.
Η τυποποίηση ενός συνόλου δεδομένων, είναι μια κοινή προ-επεξεργασία για πολλούς εκτιμητές μηχανικής μάθησης. Συνήθως αυτό γίνεται με την αφαίρεση του μέσου όρου και την κλιμάκωση στη μοναδιαία διακύμανση. Ωστόσο, οι ακραίες τιμές μπορούν συχνά να επηρεάσουν αρνητικά τη μέση τιμή/διακύμανση του δείγματος. Σε τέτοιες περιπτώσεις, η χρήση της διάμεσης και της διατεταρτημοριακής περιοχής συχνά δίνει καλύτερα αποτελέσματα.

Με την χρήση των τριών αυτών τεχνικών κλιμάκωσης χαρακτηριστικών, θα δοκιμαστεί κλιμάκωση με ευαισθησία σε ακραίες τιμές και σε μη κανονική κατανομή δεδομένων, κλιμάκωση με ευαισθησία σε ακραίες τιμές, αλλά χωρίς ευαισθησία σε μη κανονική κατανομή δεδομένων και σε κλιμάκωση ανεκτική σε ακραίες τιμές και σε μη κανονική κατανομή δεδομένων.

3. *Κανονικοποίηση δεδομένων (normalization)*

Οι τεχνικές κανονικοποίησης προσαρμόζουν την κατανομή των δεδομένων σε ένα συγκεκριμένο εύρος ή περιορισμό, όπως ένα άθροισμα απόλυτων τιμών ίσο με 1 (κανονικοποίηση L1) ή άθροισμα τετραγώνων ίσο με 1 (κανονικοποίηση L2).

i. *L1*: Κλιμακώνει τα δεδομένα, έτσι ώστε το άθροισμα των απόλυτων τιμών κάθε παρατήρησης να είναι ίσο με 1. Υπολογίζεται ως οι απόλυτες τιμές κάθε παρατήρησης, διαιρούμενες με το άθροισμα των απόλυτων τιμών. Η κανονικοποίηση L1 είναι ανθεκτική σε ακραίες τιμές, επειδή δεν βασίζεται σε τετραγωνικούς όρους. Χρησιμοποιείται συνήθως σε αλγόριθμους μηχανικής μάθησης, όπου είναι επιθυμητή η αραιότητα, όπως στην παλινδρόμηση Lasso.

ii. *L2*: Κλιμακώνει τα δεδομένα έτσι ώστε το άθροισμα των τετραγώνων κάθε παρατήρησης να είναι ίσο με 1. Υπολογίζεται ως το τετράγωνο κάθε παρατήρησης, διαιρούμενο με το άθροισμα των τετραγώνων, ακολουθούμενο από τη λήψη της τετραγωνικής ρίζας του αποτελέσματος. Η κανονικοποίηση L2 είναι ευαίσθητη σε ακραίες τιμές, επειδή τετραγωνίζει κάθε όρο. Χρησιμοποιείται συνήθως σε αλγόριθμους μηχανικής μάθησης, όπου οι μικρές διαφορές μεταξύ των παρατηρήσεων είναι σημαντικές, όπως στην παλινδρόμηση Ridge

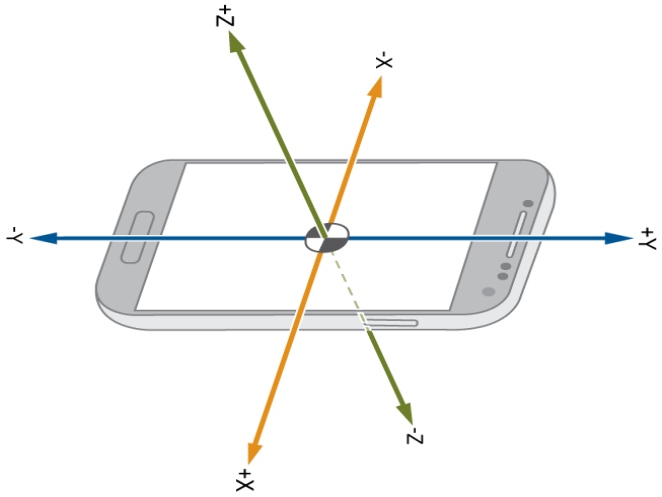
iii. *Max*: Κλιμακώνει τα δεδομένα έτσι ώστε η μέγιστη τιμή κάθε παρατήρησης να είναι ίση με 1. Υπολογίζεται διαιρώντας κάθε παρατήρηση, με τη μέγιστη τιμή στο σύνολο δεδομένων. Η μέγιστη κανονικοποίηση είναι χρήσιμη όταν το εύρος κάθε χαρακτηριστικού είναι γνωστό και η μέγιστη τιμή αντιπροσωπεύει ένα σημαντικό ανώτερο όριο. Διατηρεί την αρχική κλίμακα των δεδομένων, αλλά ενδέχεται να μην είναι ανθεκτική σε ακραίες τιμές, εάν υπάρχουν στο σύνολο δεδομένων.

Συνοπτικά, η κανονικοποίηση L1 δίνει έμφαση στη σπανιότητα και είναι ισχυρή σε ακραίες τιμές, η κανονικοποίηση L2 δίνει έμφαση στις μικρές διαφορές μεταξύ των παρατηρήσεων και είναι ευαίσθητη σε ακραίες τιμές και η κανονικοποίηση Max κλιμακώνει τα δεδομένα με βάση τη μέγιστη τιμή και διατηρεί την αρχική κλίμακα.

Δεδομένα

Για την υλοποίηση της συγκεκριμένης εφαρμογής, θα απαιτηθούν δεδομένα οδήγησης διαφορετικών οδηγών. Τα δεδομένα αυτά θα έχουν ληφθεί από αισθητήρες έξυπνων κινητών τηλεφώνων (smartphones), κατά την διάρκεια της οδήγησης. Οι αισθητήρες που απαιτούνται για τη συγκεκριμένη εφαρμογή, είναι το επιταχυνσιόμετρο (accelerator) και το γυροσκόπιο (gyroscope).

Το επιταχυνσιόμετρο χρησιμοποιείται για τη μέτρηση κραδασμών, μετατόπισης, ταχύτητας και κλίσης. Η βασική του λειτουργία που θα ληφθεί υπόψη στην εφαρμογή μας, είναι η μέτρηση της επιτάχυνσης ή της επιβράδυνσης ενός αντικειμένου.

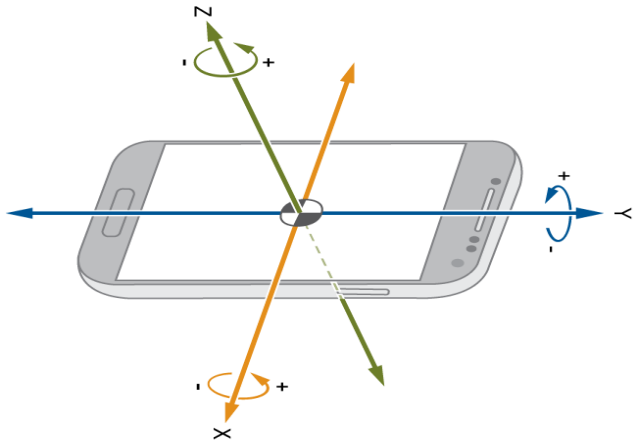


Σχήμα 28: Smartphone Accelarator Sensor Forces

Οι δυνάμεις που μετρά, αναλύονται στις τιμές τριών αξόνων (άξονες: X, Y, Z), ανιχνεύοντας τόσο το μέγεθος, όσο και την κατεύθυνση της σωστής επιτάχυνσης, ως διανυσματική ποσότητα.

Το γυροσκόπιο είναι αισθητήρας που ανιχνεύει τη γωνιακή ταχύτητα, την κατεύθυνση περιστροφής και τη γωνία περιστροφής. Αυτές θα είναι και οι βασικές λειτουργίες που θα ληφθούν υπόψη στην εφαρμογή μας, αναλύοντας πόσο ομαλές ή

απότομες είναι οι στροφές που πραγματοποιούνται κατά την οδήγηση, καθώς και οι απότομοι ελιγμοί της κατεύθυνσης του οχήματος.



Σχήμα 29: Smartphone Gyroscope Sensor Forces

Η ανίχνευση αυτών των μεγεθών επιτυγχάνεται από τις δυνάμεις που ενεργούν σε τρεις άξονες (άξονες: X, Y, Z) του αισθητήρα.
Επομένως, τα δεδομένα μας θα περιέχουν πληροφορίες για αυτούς τους αισθητήρες, ανάλογες της χρονικής στιγμής που πραγματοποιήθηκαν.

Συλλογή Δεδομένων (Data Collection)

Το σύνολο των δεδομένων (Dataset) που θα χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής, συλλέχθηκαν μέσω ενός άρθρου που έχει δημοσιευτεί στην ιστοσελίδα ScienceDirect: Smartphone Sensor Dataset for Driver Behavior Analysis by Pawan Wawage, Yogesh Deshpande, 2022
Το σύνολο των δεδομένων είναι διαθέσιμα απευθείας στον σύνδεσμο: Driver Behavior Detection Using Smartphone - Dataset

Το συμπιεσμένο αρχείο των δεδομένων που χρησιμοποιήθηκε έχει όνομα Daywise data.rar. Για λόγους ευχρηστίας, μετονομάστηκε και μετασχηματίστηκε σε **data.zip**. Μπορεί να ληφθεί από την τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

Τα δεδομένα μέσα σε αυτό το αρχείο οργανώνονται σε φακέλους, ένας για κάθε διαδρομή συγκεκριμένης μέρας. Κάθε μέρα αντιστοιχεί κατά μέσω όρο σε δύο διαδρομές. Οι διαθέσιμοι υποφάκελοι ανέρχονται σε 16.

Σχήμα 30: Υποφάκελοι Δεδομένων

Η ένδειξη "S" υποδηλώνει την διαδρομή έναρξης της συγκεκριμένης ημέρας, η ένδειξη "R" την διαδρομή επιστροφής και η ένδειξη "E" ενδιάμεσες διαδρομές. Παρατηρούμε ότι στην 5η και 7η ημέρα έχουμε μόνο διαδρομές έναρξης και όχι επιστροφής.

Οι καταγεγραμμένες παράμετροι για κάθε ταξίδι μονής διαδρομής, αποθηκεύτηκαν σε αρχεία.csv για κάθε μεμονωμένο αισθητήρα, με χρονική σήμανση που υποδεικνύει την ημερομηνία και την ώρα του ταξιδιού (Accelerometer.csv, Gyroscope.csv, κ.λπ.).

Accelerometer.csv				
Timestamp	**Milliseconds**	**X**	**Y**	**Z**
23/5/2020 13:17	2	-1,3667755	5,986679	8,086151
23/5/2020 13:17	12	-1,1716461	5,8645782	8,348328
23/5/2020 13:17	22	-0,68862915	5,6449127	8,1184845
23/5/2020 13:17	33	-0,64971924	5,6999817	7,7384033
23/5/2020 13:17	42	-0,7377014	5,8927	7,124878
23/5/2020 13:17	53	-0,8400574	6,0010376	6,813034
23/5/2020 13:17	62	-0,91007996	6,0303802	6,8651123
23/5/2020 13:17	72	-0,90530396	6,021393	7,1937103
23/5/2020 13:17	83	-0,7766113	5,9178467	7,7982483
23/5/2020 13:17	92	-0,6874237	5,816696	8,117874
23/5/2020 13:17	102	-0,5964508	5,7203217	8,304031
23/5/2020 13:17	113	-0,50128174	5,6119843	8,373459
23/5/2020 13:17	122	-0,46237183	5,563507	8,344131
23/5/2020 13:17	133	-0,5288086	5,579071	8,254944
23/5/2020 13:17	142	-0,6730652	5,622757	8,037674
23/5/2020 13:17	152	-0,7742157	5,662857	7,9018097

Σχήμα 31: Δεδομένα Accelarator.csv

Gyroscope.csv				
Timestamp	Milliseconds	X	Y	Z
23/5/2020 13:17	3	-0,18132019	-0,041305542	0,03315735
23/5/2020 13:17	12	0,08979797	0,20851135	0,121032715
23/5/2020 13:17	22	0,14891052	0,19998169	0,10293579
23/5/2020 13:17	31	0,15904236	0,15631104	0,071502686
23/5/2020 13:17	41	0,10044861	0,06629944	0,033691406
23/5/2020 13:17	52	-0,019927979	0,007171631	0,031021118
23/5/2020 13:17	61	-0,10835266	-0,001876831	0,04647827
23/5/2020 13:17	71	-0,19729614	7,78E-04	0,07896423
23/5/2020 13:17	82	-0,21487427	0,008239746	0,0960083
23/5/2020 13:17	91	-0,1829071	0,014633179	0,104522705
23/5/2020 13:17	102	-0,1424408	0,016220093	0,101867676
23/5/2020 13:17	111	-0,07159424	0,030075073	0,08854675
23/5/2020 13:17	122	-0,020996094	0,045516968	0,07575989
23/5/2020 13:17	132	0,02748108	0,06097412	0,06298828
23/5/2020 13:17	141	0,077545166	0,081207275	0,050201416
23/5/2020 13:17	152	0,077545166	0,08067322	0,052856445

Σχήμα 32: Δεδομένα Gyroscope.csv

Στα παραπάνω σχήματα, εμφανίζονται τα πεδία των δεδομένων ανά αισθητήρα.

Επίσης κάποιες πληροφορίες που αφορούν τα δεδομένα είναι:

- Οι δραστηριότητες οδήγησης έλαβαν χώρα σε ποικίλες οδικές και κυκλοφοριακές συνθήκες (πρωί, μεσημέρι, απόγευμα, βράδυ).
- Το τηλέφωνο είναι στερεωμένο οριζοντίως κατά την διάρκεια της μέτρησης.
- Για τη διεξαγωγή των δραστηριοτήτων οδήγησης και τη συλλογή δεδομένων χρησιμοποιήθηκαν διάφορα οχήματα (τετράτροχα) και κινητά τηλέφωνα.
- Την οδήγηση πραγματοποίησαν επτά οδηγοί, ένας για κάθε ημέρα.
- Στο σύνολο των δεδομένων περιέχονται περίπου 15.000+ τιμές αισθητήρων για κάθε μονόδρομο ταξίδι 5–17 χιλιομέτρων.

Για τις μονάδες μέτρησης των δεδομένων των αισθητήρων αναφέρεται ότι: το επιταχυνσιόμετρο (άξονας X, Y, Z σε μέτρα ανά δευτερόλεπτο στο τετράγωνο (m/s2)) και το γυροσκόπιο (άξονας X, Y, Z σε μοίρες ανά δευτερόλεπτο (°/s)).

Για την προ-επεξεργασία και μετασχηματισμό των δεδομένων, έτσι ώστε να είναι σε μορφή αξιοποιήσιμη από την εφαρμογή, έχει δημιουργηθεί κώδικας στην γλώσσα προγραμματισμού Python με όνομα **preprocessData.py,** το οποίο μπορεί να ληφθεί

από την τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

Προ-επεξεργασία και Μετασχηματισμός Δεδομένων

Στην διαδικασία αυτή διενεργήθηκε εύρεση και τακτοποίηση λαθών και ελλείψεων των δεδομένων, με χρήση συγκεκριμένων διαδικασιών. Επίσης έγινε κανονικοποίηση των δεδομένων σε κοινή μορφή, η οποία περιλαμβάνει εξάλειψη τυχόν περιττών δεδομένων, σύμπτυξη δεδομένων, εξάλειψη θορύβου, δημιουργία νέων δεδομένων βασισμένων στα ήδη υπάρχοντα.

Για την υλοποίηση των διαδικασιών, χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες της Python:

```
from pathlib import Path
import pandas as pd
import os
import sys
import zipfile
from datetime import datetime
import numpy as np
```

Σχήμα 33: Βιβλιοθήκες Python για προ-επεξεργασία δεδομένων

Οι βιβλιοθήκες ***Path*** και ***os*** χρησιμοποιήθηκαν για τον εντοπισμό των αρχείων δεδομένων.

Η βιβλιοθήκη ***datetime*** για την επεξεργασία ημερομηνιών

Η βιβλιοθήκη ***zipfile*** για την αποσυμπίεση αρχείων μορφής .zip

Η βιβλιοθήκη ***pandas*** για την επεξεργασία των dataframes των δεδομένων

Η βιβλιοθήκη ***numpy*** για την διαχείριση των τύπων των δεδομένων των dataframes.

Παρατηρήθηκε ότι τα δεδομένα είναι συμπιεσμένα σε ένα αρχείο. Το αρχείο αυτό τοποθετείται στην C:\datasets (αν δεν υπάρχει το δημιουργούμε).

Η πρώτη εργασία είναι η αποσυμπίεση των δεδομένων. Αυτό επιτυγχάνεται μέσω του κώδικα:

```python
def extract_zip(zip_filename, extract_path):
    try:
        with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
            zip_ref.extractall(extract_path)
        print("Extraction completed successfully.")
    except zipfile.BadZipFile:
        print("Error: File is not a valid ZIP file.")
    except FileNotFoundError as e:
        print(f"Error: {e}")
        print(f"File not found: {zip_filename}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

Παράρτημα Κώδικα 1: Συνάρτηση Αποσυμπίεσης Αρχείου Δεδομένων

```python
if not os.path.isdir('C:\datasets\data'):
    zip_filename = "C:\datasets\data.zip"
    extract_path = "C:\datasets"
    extract_zip(zip_filename, extract_path)
```

Παράρτημα Κώδικα 2: Ανάκληση συνάρτησης αποσυμπίεσης

Παρατηρήθηκε ότι τα δεδομένα για τον κάθε αισθητήρα, βρίσκονται σε διαφορετικό αρχείο csv μέσα στον ίδιο φάκελο. Επομένως πραγματοποιήθηκε συνένωση των δύο αρχείων csv. Για να γίνει αυτό αντλήθηκαν τα δεδομένα του κάθε αισθητήρα, σε ξεχωριστό dataframe. Μετονομάστηκαν τα πεδία των τιμών τους X, Y, Z σε AccX, AccY, AccZ για το accelerator και GyroX, GyroY, GyroZ για το gyroscope. Τα πεδία Timestamp και Milliseconds είναι κοινά και για τα δύο αρχεία, αλλά μετονομάστηκαν σε Date και Ms για ευχρηστία και καλύτερη αντίληψη. Επίσης ορίστηκαν διαφορετικοί τύποι δεδομένων από τους εξ ορισμού της python (πιο μικροί), για εξοικονόμηση μνήμης:

Date: Από Object σε Datetime[ns],

Ms: Από int64 σε int32,

AccX, AccY, AccZ, GyroX, GyroY, GyroZ: Από float64 σε float32

Στο κάθε dataframe δημιουργήθηκαν αντίστοιχα κενά πεδία του άλλου dataframe, έτσι ώστε να είναι κοινή η γραμμογράφησή τους και να είναι ευκολότερη η συνένωσή τους (δηλαδή στο dataframe του accelerator τα πεδία GyroX, GyroY, GyroZ και στο gyroscope AccX, AccY, AccZ.). Επομένως το κάθε dataframe έχει τα κοινά πεδία:

Date, Ms, AccX, AccY, AccZ, GyroX, GyroY, GyroZ

Επίσης, για την ημέρα που αφορούν τα δεδομένα καθώς και το εύρος ώρας που διενεργήθηκε η κάθε διαδρομή, δημιουργήθηκαν το πεδίο Driver για τον χαρακτηρισμό του οδηγού που διενέργησε την οδήγηση, δηλώνοντας για την κάθε ημέρα / οδηγό αντίστοιχη κωδικοποίηση (1, 2.., 7) και το πεδίο DayPart για τη οριοθέτηση του μέρους της ημέρας που πραγματοποιήθηκε η διαδρομή, με αντίστοιχη κωδικοποίηση που αντιπροσωπεύει τα Πρωί, Μεσημέρι, Απόγευμα, Βράδυ (0.25 για το χρονικό διάστημα 05:00:00 – 11:59:59, 0.5 για το χρονικό διάστημα 12:00:00 – 16:59:59, 0.75 για το χρονικό διάστημα 17:00:00 – 20:59:59, 1.0 για το χρονικό διάστημα 21:00:00 – 23:59:59). Επίσης, δημιουργήθηκε το πεδίο με όνομα Millsecs, το οποίο αντιπροσωπεύει την ημερομηνία με την ώρα σε milliseconds, στην οποία έχει προστεθεί η στήλη Ms με τα επιπλέον milliseconds. Αποτέλεσμα είναι τα συνολικά milliseconds που αντιστοιχούν στην εγγραφή. Οι διαδικασίες αυτές εκτελούνται μέσω της συνάρτησης load_data

```
def load_data(sensor, file):
    if sensor=="Accelerometer":
        colnames = ['Date', 'Ms', 'AccX', 'AccY', 'AccZ']
        datatypes = {'Ms': 'int32', 'AccX': 'float32', 'AccY': 'float32','AccZ': 'float32'}
    elif sensor=="Gyroscope":
        colnames = ['Date', 'Ms', 'GyroX', 'GyroY', 'GyroZ']
        datatypes = {'Ms': 'int32', 'GyroX': 'float32', 'GyroY': 'float32','GyroZ': 'float32'}

    dfTemp = pd.read_csv(file, header=0, names=colnames, dtype=datatypes, parse_dates=['Date'], dayfirst=True)

    timeStart = dfTemp['Date'].iloc[0].time()
    timeEnd = dfTemp['Date'].iloc[len(dfTemp.index) - 1].time()
    driveTime = check_time(timeStart, timeEnd)
    dfTemp.insert(loc=0, column='DayPart', value=driveTime)

    pos = str(file).find("Day-")
    driver = int(str(file)[ pos + 4 : pos + 5])
    dfTemp.insert(loc=0, column='Driver', value=driver)
    dfTemp['Date'] = dfTemp['Date'].astype('datetime64[ns]')
    df1=(dfTemp['Date'].astype(np.int64) // 10**6) + dfTemp['Ms']
    dfTemp.insert(loc=4, column='Millsecs', value=df1)
    #dfTemp['Date'] = dfTemp['Date'].apply(lambda x: x.strftime('%Y-%m-%d'))

    if sensor=="Accelerometer":
        dfTemp['GyroX']=np.nan
        dfTemp['GyroY']=np.nan
        dfTemp['GyroZ']=np.nan
    elif sensor=="Gyroscope":
        dfTemp.insert(loc=4, column='AccZ', value=np.nan)
        dfTemp.insert(loc=4, column='AccY', value=np.nan)
        dfTemp.insert(loc=4, column='AccX', value=np.nan)
    return dfTemp
```

Παράρτημα Κώδικα 3: Συνάρτηση Προ-επεξεργασίας Δεδομένων Αισθητήρων

Μέσα στην συνάρτηση load_data καλείται η συνάρτηση check_time, η οποία επιστρέφει κωδικοποιημένο το μέρος της ημέρας.

```
MORNING_START   = datetime.strptime('05::00::00', '%H::%M::%S').time()
MORNING_END     = datetime.strptime('11::59::59', '%H::%M::%S').time()
AFTERNOON_START = datetime.strptime('12::00::00', '%H::%M::%S').time()
AFTERNOON_END   = datetime.strptime('16::59::59', '%H::%M::%S').time()
EVENING_START   = datetime.strptime('17::00::00', '%H::%M::%S').time()
EVENING_END     = datetime.strptime('20::59::59', '%H::%M::%S').time()
NIGHT_START     = datetime.strptime('21::00::00', '%H::%M::%S').time()
NIGHT_END       = datetime.strptime('23::59::59', '%H::%M::%S').time()

def check_time(timeStart, timeEnd):
    if timeStart >= MORNING_START and timeEnd <= MORNING_END:
        return 0.25
    elif timeStart >= AFTERNOON_START and timeEnd <= AFTERNOON_END:
        return 0.50
    elif timeStart >= EVENING_START and timeEnd <= EVENING_END:
        return 0.75
    elif timeStart >= NIGHT_START and timeEnd <= NIGHT_END:
        return 1.00
    else:
        return '?'
```

Παράρτημα Κώδικα 4: Συνάρτηση για το χρονικό διάστημα της ημέρας

Η συνάρτηση load_data καλείται για τον κάθε αισθητήρα για κάθε διαδρομή:

```
sensors=['Accelerometer', 'Gyroscope']
df_list = list()
i=1
folders_list = os.listdir(Path("c:/datasets/data"))

for x in folders_list:
    dfItem_list = list()
    print("\nLoad Sensors data from c:/datasets/data/"+x+" ("+str(i)+" of "+str(len(folders_list))+")...\n")
    for sensor in sensors:
        file=Path("c:/datasets/data/"+x+"/"+sensor+".csv")
        if file.exists():
            print("FIle "+x+"/"+sensor+".csv...")
            dfTemp=load_data(sensor, file)
            dfItem_list.append(dfTemp)
            del dfTemp
```

Παράρτημα Κώδικα 5: Ανάκληση Συνάρτησης Προ-επεξεργασίας Δεδομένων Αισθητήρων

Για κάθε ζεύγος αισθητήρων διενεργείται συνένωση (concat), η οποία συνενώνει τα δύο dataframes σε ένα (το ένα κάτω από το άλλο). Επιλέχθηκε αυτή η διαδικασία γιατί παρατηρήθηκε ότι ενώ τα πεδία Date και Ms είναι κοινά, δεν υπάρχει απόλυτη αντιστοίχιση μεταξύ τους, όπως φαίνεται για παράδειγμα στα Σχήματα 31 και 32 όπου η πρώτη εγγραφή και για τα δύο έχουν διαφορετική τιμή Ms. Αντίστοιχα μπορεί να διαφέρει και η ώρα. Αυτό συμβαίνει διότι ο κάθε αισθητήρας μπορεί να παράγει στιγμιότυπα σε διαφορετικές χρονικές στιγμές. Με αυτό τον τρόπο δεν χάνουμε δεδομένα από κανένα αισθητήρα αν και δημιουργούνται κενά τα οποία διαχειριζόμαστε σε επόμενο βήμα.

	Driver	DayPart	Date	Ms	Millsecs	AccX	AccY	AccZ	GyroX	GyroY	GyroZ
0	1	0.5	2020-05-23 13:17:00	2	1590239820002	-1.366776	5.986679	8.086151	NaN	NaN	NaN
1	1	0.5	2020-05-23 13:17:00	3	1590239820003	NaN	NaN	NaN	-0.181320	-0.041306	0.033157
2	1	0.5	2020-05-23 13:17:00	12	1590239820012	-1.171646	5.864578	8.348328	NaN	NaN	NaN
3	1	0.5	2020-05-23 13:17:00	12	1590239820012	NaN	NaN	NaN	0.089798	0.208511	0.121033
4	1	0.5	2020-05-23 13:17:00	22	1590239820022	-0.688629	5.644913	8.118484	NaN	NaN	NaN
5	1	0.5	2020-05-23 13:17:00	22	1590239820022	NaN	NaN	NaN	0.148911	0.199982	0.102936
6	1	0.5	2020-05-23 13:17:00	31	1590239820031	NaN	NaN	NaN	0.159042	0.156311	0.071503
7	1	0.5	2020-05-23 13:17:00	33	1590239820033	-0.649719	5.699982	7.738403	NaN	NaN	NaN
8	1	0.5	2020-05-23 13:17:00	41	1590239820041	NaN	NaN	NaN	0.100449	0.066299	0.033691
9	1	0.5	2020-05-23 13:17:00	42	1590239820042	-0.737701	5.892700	7.124878	NaN	NaN	NaN
10	1	0.5	2020-05-23 13:17:00	52	1590239820052	NaN	NaN	NaN	-0.019928	0.007172	0.031021
11	1	0.5	2020-05-23 13:17:00	53	1590239820053	-0.840057	6.001038	6.813034	NaN	NaN	NaN
12	1	0.5	2020-05-23 13:17:00	61	1590239820061	NaN	NaN	NaN	-0.108353	-0.001877	0.046478
13	1	0.5	2020-05-23 13:17:00	62	1590239820062	-0.910080	6.030380	6.865112	NaN	NaN	NaN
14	1	0.5	2020-05-23 13:17:00	71	1590239820071	NaN	NaN	NaN	-0.197296	0.000778	0.078964
15	1	0.5	2020-05-23 13:17:00	72	1590239820072	-0.905304	6.021393	7.193710	NaN	NaN	NaN
16	1	0.5	2020-05-23 13:17:00	82	1590239820082	NaN	NaN	NaN	-0.214874	0.008240	0.096008
17	1	0.5	2020-05-23 13:17:00	83	1590239820083	-0.776611	5.917847	7.798248	NaN	NaN	NaN
18	1	0.5	2020-05-23 13:17:00	91	1590239820091	NaN	NaN	NaN	-0.182907	0.014633	0.104523
19	1	0.5	2020-05-23 13:17:00	92	1590239820092	-0.687424	5.816696	8.117874	NaN	NaN	NaN

Παράρτημα Δεδομένων 1: Συνένωση δεδομένων αισθητήρων

Η εκτέλεση της συνένωσης έγινε ως εξής:

```
df = pd.concat([dfItem_list[0], dfItem_list[1]])
```

Παράρτημα Κώδικα 6: Συνένωση δεδομένων Αισθητήρων

Και η λειτουργία της ταξινόμησης:

```
df = df.sort_values(by=["Driver", "DayPart", "Date","Ms"], ignore_index=True)
```

Παράρτημα Κώδικα 7: Ταξινόμηση Συνενωμένων Δεδομένων

Έπειτα πραγματοποιήθηκε ομαδοποίηση των δεδομένων της ίδιας χρονικής στιγμής και για τους δύο αισθητήρες:

```
dfGroup = df.groupby(["Driver", "DayPart", 'Date', 'Ms'], as_index=False)
df=dfGroup.max()
```

Παράρτημα Κώδικα 8: Ομαδοποίηση κοινών δεδομένων

Το αποτέλεσμα της ομαδοποίησης:

	Driver	DayPart	Date	Ms	Millsecs	AccX	AccY	AccZ	GyroX	GyroY	GyroZ
0	1	0.5	2020-05-23 13:17:00	2	1590239820002	-1.366776	5.986679	8.086151	NaN	NaN	NaN
1	1	0.5	2020-05-23 13:17:00	3	1590239820003	NaN	NaN	NaN	-0.181320	-0.041306	0.033157
2	1	0.5	2020-05-23 13:17:00	12	1590239820012	-1.171646	5.864578	8.348328	0.089798	0.208511	0.121033
3	1	0.5	2020-05-23 13:17:00	22	1590239820022	-0.688629	5.644913	8.118484	0.148911	0.199982	0.102936
4	1	0.5	2020-05-23 13:17:00	31	1590239820031	NaN	NaN	NaN	0.159042	0.156311	0.071503
5	1	0.5	2020-05-23 13:17:00	33	1590239820033	-0.649719	5.699982	7.738403	NaN	NaN	NaN
6	1	0.5	2020-05-23 13:17:00	41	1590239820041	NaN	NaN	NaN	0.100449	0.066299	0.033691
7	1	0.5	2020-05-23 13:17:00	42	1590239820042	-0.737701	5.892700	7.124878	NaN	NaN	NaN
8	1	0.5	2020-05-23 13:17:00	52	1590239820052	NaN	NaN	NaN	-0.019928	0.007172	0.031021
9	1	0.5	2020-05-23 13:17:00	53	1590239820053	-0.840057	6.001038	6.813034	NaN	NaN	NaN
10	1	0.5	2020-05-23 13:17:00	61	1590239820061	NaN	NaN	NaN	-0.108353	-0.001877	0.046478
11	1	0.5	2020-05-23 13:17:00	62	1590239820062	-0.910080	6.030380	6.865112	NaN	NaN	NaN
12	1	0.5	2020-05-23 13:17:00	71	1590239820071	NaN	NaN	NaN	-0.197296	0.000778	0.078964
13	1	0.5	2020-05-23 13:17:00	72	1590239820072	-0.905304	6.021393	7.193710	NaN	NaN	NaN
14	1	0.5	2020-05-23 13:17:00	82	1590239820082	NaN	NaN	NaN	-0.214874	0.008240	0.096008
15	1	0.5	2020-05-23 13:17:00	83	1590239820083	-0.776611	5.917847	7.798248	NaN	NaN	NaN
16	1	0.5	2020-05-23 13:17:00	91	1590239820091	NaN	NaN	NaN	-0.182907	0.014633	0.104523
17	1	0.5	2020-05-23 13:17:00	92	1590239820092	-0.687424	5.816696	8.117874	NaN	NaN	NaN
18	1	0.5	2020-05-23 13:17:00	102	1590239820102	-0.596451	5.720322	8.304031	-0.142441	0.016220	0.101868
19	1	0.5	2020-05-23 13:17:00	111	1590239820111	NaN	NaN	NaN	-0.071594	0.030075	0.088547

Παράρτημα Δεδομένων 2: Ομαδοποίηση Δεδομένων Συνένωσης

Παρατηρούμε ότι έχουν μείνει κενές οι τιμές, για τις οποίες δεν υπάρχει κοινή χρονική στιγμή. Αποφασίστηκε η ενημέρωση αυτών των τιμών με την προηγούμενη τιμή του αντίστοιχου πεδίου μιας και είναι η τιμή της προηγούμενης από αυτήν χρονικής στιγμής. Αυτό επιτυγχάνεται με τον κώδικα:

```
cols = ['Driver', 'DayPart', 'AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']
df.loc[:,cols] = df.loc[:,cols].ffill()
```

Παράρτημα Κώδικα 9: Γέμισμα κενών τιμών με την προηγούμενη του πεδίου

Το αποτέλεσμα:

	Driver	DayPart	Date	Ms	Millsecs	AccX	AccY	AccZ	GyroX	GyroY	GyroZ
0	1	0.5	2020-05-23 13:17:00	2	1590239820002	-1.366776	5.986679	8.086151	NaN	NaN	NaN
1	1	0.5	2020-05-23 13:17:00	3	1590239820003	-1.366776	5.986679	8.086151	-0.181320	-0.041306	0.033157
2	1	0.5	2020-05-23 13:17:00	12	1590239820012	-1.171646	5.864578	8.348328	0.089798	0.208511	0.121033
3	1	0.5	2020-05-23 13:17:00	22	1590239820022	-0.688629	5.644913	8.118484	0.148911	0.199982	0.102936
4	1	0.5	2020-05-23 13:17:00	31	1590239820031	-0.688629	5.644913	8.118484	0.159042	0.156311	0.071503
5	1	0.5	2020-05-23 13:17:00	33	1590239820033	-0.649719	5.699982	7.738403	0.159042	0.156311	0.071503
6	1	0.5	2020-05-23 13:17:00	41	1590239820041	-0.649719	5.699982	7.738403	0.100449	0.066299	0.033691
7	1	0.5	2020-05-23 13:17:00	42	1590239820042	-0.737701	5.892700	7.124878	0.100449	0.066299	0.033691
8	1	0.5	2020-05-23 13:17:00	52	1590239820052	-0.737701	5.892700	7.124878	-0.019928	0.007172	0.031021
9	1	0.5	2020-05-23 13:17:00	53	1590239820053	-0.840057	6.001038	6.813034	-0.019928	0.007172	0.031021
10	1	0.5	2020-05-23 13:17:00	61	1590239820061	-0.840057	6.001038	6.813034	-0.108353	-0.001877	0.046478
11	1	0.5	2020-05-23 13:17:00	62	1590239820062	-0.910080	6.030380	6.865112	-0.108353	-0.001877	0.046478
12	1	0.5	2020-05-23 13:17:00	71	1590239820071	-0.910080	6.030380	6.865112	-0.197296	0.000778	0.078964
13	1	0.5	2020-05-23 13:17:00	72	1590239820072	-0.905304	6.021393	7.193710	-0.197296	0.000778	0.078964
14	1	0.5	2020-05-23 13:17:00	82	1590239820082	-0.905304	6.021393	7.193710	-0.214874	0.008240	0.096008
15	1	0.5	2020-05-23 13:17:00	83	1590239820083	-0.776611	5.917847	7.798248	-0.214874	0.008240	0.096008
16	1	0.5	2020-05-23 13:17:00	91	1590239820091	-0.776611	5.917847	7.798248	-0.182907	0.014633	0.104523
17	1	0.5	2020-05-23 13:17:00	92	1590239820092	-0.687424	5.816696	8.117874	-0.182907	0.014633	0.104523
18	1	0.5	2020-05-23 13:17:00	102	1590239820102	-0.596451	5.720322	8.304031	-0.142441	0.016220	0.101868
19	1	0.5	2020-05-23 13:17:00	111	1590239820111	-0.596451	5.720322	8.304031	-0.071594	0.030075	0.088547

Παράρτημα Δεδομένων 3: Γέμισμα κενών τιμών με την προηγούμενη του πεδίου

Στο αποτέλεσμα δεν επηρεάζονται όπως είναι φυσικό οι τιμές της πρώτης εγγραφής, για τις οποίες αποφασίστηκε να ενημερωθούν από τις επόμενες τιμές μιας και η εκκίνηση της καταγραφής θα πρέπει να έχει τις πρώτες ενδείξεις των αισθητήρων.

Αυτό πραγματοποιήθηκε με τον παρακάτω κώδικα:

```
df.loc[:,cols] = df.loc[:,cols].bfill()
```

Παράρτημα Κώδικα 10: Γέμισμα κενών τιμών με την επόμενη του πεδίου

Το αποτέλεσμα:

	Driver	DayPart	Date	Ms	Millsecs	AccX	AccY	AccZ	GyroX	GyroY	GyroZ
0	1	0.5	2020-05-23 13:17:00	2	1590239820002	-1.366776	5.986679	8.086151	-0.181320	-0.041306	0.033157
1	1	0.5	2020-05-23 13:17:00	3	1590239820003	-1.366776	5.986679	8.086151	-0.181320	-0.041306	0.033157
2	1	0.5	2020-05-23 13:17:00	12	1590239820012	-1.171646	5.864578	8.348328	0.089798	0.208511	0.121033
3	1	0.5	2020-05-23 13:17:00	22	1590239820022	-0.688629	5.644913	8.118484	0.148911	0.199982	0.102936
4	1	0.5	2020-05-23 13:17:00	31	1590239820031	-0.688629	5.644913	8.118484	0.159042	0.156311	0.071503
5	1	0.5	2020-05-23 13:17:00	33	1590239820033	-0.649719	5.699982	7.738403	0.159042	0.156311	0.071503
6	1	0.5	2020-05-23 13:17:00	41	1590239820041	-0.649719	5.699982	7.738403	0.100449	0.066299	0.033691
7	1	0.5	2020-05-23 13:17:00	42	1590239820042	-0.737701	5.892700	7.124878	0.100449	0.066299	0.033691
8	1	0.5	2020-05-23 13:17:00	52	1590239820052	-0.737701	5.892700	7.124878	-0.019928	0.007172	0.031021
9	1	0.5	2020-05-23 13:17:00	53	1590239820053	-0.840057	6.001038	6.813034	-0.019928	0.007172	0.031021
10	1	0.5	2020-05-23 13:17:00	61	1590239820061	-0.840057	6.001038	6.813034	-0.108353	-0.001877	0.046478
11	1	0.5	2020-05-23 13:17:00	62	1590239820062	-0.910080	6.030380	6.865112	-0.108353	-0.001877	0.046478
12	1	0.5	2020-05-23 13:17:00	71	1590239820071	-0.910080	6.030380	6.865112	-0.197296	0.000778	0.078964
13	1	0.5	2020-05-23 13:17:00	72	1590239820072	-0.905304	6.021393	7.193710	-0.197296	0.000778	0.078964
14	1	0.5	2020-05-23 13:17:00	82	1590239820082	-0.905304	6.021393	7.193710	-0.214874	0.008240	0.096008
15	1	0.5	2020-05-23 13:17:00	83	1590239820083	-0.776611	5.917847	7.798248	-0.214874	0.008240	0.096008
16	1	0.5	2020-05-23 13:17:00	91	1590239820091	-0.776611	5.917847	7.798248	-0.182907	0.014633	0.104523
17	1	0.5	2020-05-23 13:17:00	92	1590239820092	-0.687424	5.816696	8.117874	-0.182907	0.014633	0.104523
18	1	0.5	2020-05-23 13:17:00	102	1590239820102	-0.596451	5.720322	8.304031	-0.142441	0.016220	0.101868
19	1	0.5	2020-05-23 13:17:00	111	1590239820111	-0.596451	5.720322	8.304031	-0.071594	0.030075	0.088547

Παράρτημα Δεδομένων 4: Γέμισμα κενών τιμών με την επόμενη του πεδίου

Επίσης παρατηρήθηκε ότι στα εισαγωγικά δεδομένα, μπορεί να υπάρξουν διπλότυπες εγγραφές, που αποτυπώνουν ενδείξεις των αισθητήρων για την ίδια ημέρα για την ίδια διαδρομή και για την ίδια χρονική στιγμή, οι οποίες ενδείξεις είναι πανομοιότυπες. Αποφασίστηκε η εξάλειψη αυτών των τιμών, διότι δεν συνεισφέρουν στην ορθολογική αποτίμηση της πληροφορίας που θέλουμε να παράγουμε. Παράδειγμα τέτοιων δεδομένων φαίνεται στο παρακάτω παράρτημα:

Timestamp	Milliseconds	X	Y	Z
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209396	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656
23/5/2020 7:49	209397	3,5916138	0,796051	9,944656

Παράρτημα Δεδομένων 5: Διπλότυπες Εγγραφές Αισθητήρα

Η διαδικασία αυτή εκτελέστηκε με τον παρακάτω κώδικα:

```
df.drop_duplicates(keep='first', inplace=True, ignore_index=True)
```

Παράρτημα Κώδικα 11: Εξάλειψη διπλότυπων εγγραφών

Τέλος το κάθε ζεύγος αισθητήρων που ενώθηκε, καταχωρείται σε μια λίστα dataframes τα οποία στην συνέχεια συνενώνονται σε ένα dataframe. Αυτό είναι το τελικό dataframe, το οποίο εξάγεται ως αρχείο csv με όνομα **finalData.csv,** συμπιεσμένο με μορφή .zip και όνομα **finalData.csv.zip**. Το αρχείο αυτό, περιέχει τα δεδομένα που θα χρησιμοποιηθούν από την εφαρμογή μας και μπορεί να ληφθεί από την τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

Διερευνητική Ανάλυση Δεδομένων

Πριν την εφαρμογή του αλγόριθμου συσταδοποίησης, απαραίτητο βήμα είναι να πραγματοποιηθεί διερευνητική ανάλυση των δεδομένων (Exploratory Data Analysis - EDA)**,** ένα κρίσιμο βήμα στη διαδικασία ανάλυσης δεδομένων, που περιλαμβάνει την εξερεύνηση και την κατανόηση των χαρακτηριστικών ενός συνόλου. Το EDA, βοηθά στην απόκτηση γνώσεων για τα δεδομένα, τον εντοπισμό προτύπων, τον εντοπισμό ανωμαλιών και τη διατύπωση υποθέσεων.[23]

Επιθεώρηση Δεδομένων

Πρωταρχικό βήμα είναι η παρατήρηση του συνόλου των δεδομένων, έτσι ώστε να γίνει κατανοητή η δομή, το μέγεθος και τη μορφή του. Αυτό περιλαμβάνει τον έλεγχο του αριθμού των γραμμών και στηλών, των τύπων δεδομένων κάθε στήλης, της παρουσίας τιμών που λείπουν και διπλότυπων εγγραφών.

First 20 rows of the dataset:

	Driver	DayPart	Date	Ms	Millsecs	AccX	AccY	AccZ	GyroX	GyroY	GyroZ
0	1	0.25	2020-05-23 07:46:00+03:00	1	1590209160001	-0.005081	3.642151	9.712418	-0.502960	-0.306381	-0.002258
1	1	0.25	2020-05-23 07:46:00+03:00	5	1590209160005	-0.005081	3.642151	9.712418	-0.502960	-0.306381	-0.002258
2	1	0.25	2020-05-23 07:46:00+03:00	12	1590209160012	-0.005081	3.642151	9.712418	-0.262756	-0.094391	0.036621
3	1	0.25	2020-05-23 07:46:00+03:00	13	1590209160013	-0.176865	3.452423	9.970398	-0.262756	-0.094391	0.036621
4	1	0.25	2020-05-23 07:46:00+03:00	22	1590209160022	-0.176865	3.452423	9.970398	0.199036	0.248093	0.095215
5	1	0.25	2020-05-23 07:46:00+03:00	23	1590209160023	-0.067932	3.423096	9.891983	0.199036	0.248093	0.095215
6	1	0.25	2020-05-23 07:46:00+03:00	31	1590209160031	-0.067932	3.423096	9.891983	0.287460	0.285370	0.090942
7	1	0.25	2020-05-23 07:46:00+03:00	32	1590209160032	0.134979	3.409927	9.470001	0.287460	0.285370	0.090942
8	1	0.25	2020-05-23 07:46:00+03:00	42	1590209160042	0.205612	3.350662	8.900192	0.254440	0.248093	0.054199
9	1	0.25	2020-05-23 07:46:00+03:00	51	1590209160051	0.205612	3.350662	8.900192	0.145248	0.185242	0.032898
10	1	0.25	2020-05-23 07:46:00+03:00	52	1590209160052	-0.001480	3.497910	8.071198	0.145248	0.185242	0.032898
11	1	0.25	2020-05-23 07:46:00+03:00	62	1590209160062	-0.001480	3.497910	8.071198	-0.074203	0.068054	0.029694
12	1	0.25	2020-05-23 07:46:00+03:00	63	1590209160063	-0.124191	3.609833	8.047852	-0.074203	0.068054	0.029694
13	1	0.25	2020-05-23 07:46:00+03:00	71	1590209160071	-0.124191	3.609833	8.047852	-0.206284	-0.001190	0.045670
14	1	0.25	2020-05-23 07:46:00+03:00	72	1590209160072	-0.225342	3.701416	8.368668	-0.206284	-0.001190	0.045670
15	1	0.25	2020-05-23 07:46:00+03:00	82	1590209160082	-0.182251	3.619415	8.748154	-0.319748	-0.049118	0.080826
16	1	0.25	2020-05-23 07:46:00+03:00	91	1590209160091	-0.182251	3.619415	8.748154	-0.320801	-0.037933	0.098404
17	1	0.25	2020-05-23 07:46:00+03:00	93	1590209160093	0.034424	3.447632	9.303604	-0.320801	-0.037933	0.098404
18	1	0.25	2020-05-23 07:46:00+03:00	102	1590209160102	0.193039	3.326126	9.572357	-0.232391	0.027039	0.102127
19	1	0.25	2020-05-23 07:46:00+03:00	111	1590209160111	0.193039	3.326126	9.572357	-0.137573	0.085098	0.092010

Παράρτημα Δεδομένων 6: Πρώτες 20 εγγραφές του τελικού Dataset

Όπως παρατηρούμε από το παραπάνω παράρτημα, τα δεδομένα μας αποτελούνται από 11 χαρακτηριστικά. Το χαρακτηριστικό ‘Driver’ αντιπροσωπεύει τον οδηγό-ημέρα που διετελέστηκε η κάθε διαδρομή και το ‘DayPart’ το διάστημα της ημέρας. Το χαρακτηριστικό ‘Date’ αντιπροσωπεύει την τοπική ημέρα-ώρα που διετελέστηκε η κάθε διαδρομή και το ‘Ms’ τα milliseconds μέσα στην ημέρα-ώρα που ελήφθησαν τα δεδομένα των αισθητήρων. Το χαρακτηριστικό ‘Millsecs’ αντιπροσωπεύει τα συνολικά milliseconds που αποτελούνται τα ‘Date’ και ‘Ms’. Τα χαρακτηριστικά ‘AccX’, ‘AccY’, ‘AccZ’ αντιπροσωπεύουν τα δεδομένα του αισθητήρα του επιταχυνσιόμετρου που ελήφθησαν την δεδομένη χρονική στιγμή, καθώς και τα ‘GyroX’, ‘GyroY’, ‘GyroZ’ αντιπροσωπεύουν τα δεδομένα του αισθητήρα του γυροσκοπίου αντίστοιχα. Τα δεδομένα των αισθητήρων περιέχουν αρνητικές τιμές. Το dataset αποτελείται από 178360 εγγραφές.

```
Data types of columns:
 Driver        int64
DayPart      float64
Date          object
Ms             int64
Millsecs       int64
AccX         float64
AccY         float64
AccZ         float64
GyroX        float64
GyroY        float64
GyroZ        float64
dtype: object

Missing values in the dataset:
 Driver      0
DayPart      0
Date         0
Ms           0
Millsecs     0
AccX         0
AccY         0
AccZ         0
GyroX        0
GyroY        0
GyroZ        0
dtype: int64

Duplicate rows in the dataset:
 Empty DataFrame
Columns: [Driver, DayPart, Date, Ms, Millsecs, AccX, AccY, AccZ, GyroX, GyroY, GyroZ]
Index: []
```

Παράρτημα Δεδομένων 7: Δομή του τελικού Dataset

Από το παραπάνω παράρτημα παρατηρούμε ότι τα χαρακτηριστικά του dataset είναι σχεδόν όλα τύποι αριθμητικών τιμών (3 int64, 7 float64), εκτός από το 'Date' που είναι τύπος object, γεγονός που διευκολύνει την χρησιμότητα στην ενδεχόμενη συμμετοχή τους στην συσταδοποίηση. Κανένα από τα χαρακτηριστικά δεν έχει κενές τιμές, γεγονός που θα είναι χρήσιμο για εξαγωγή πιο αξιόπιστων αποτελεσμάτων, μιας και πολλοί αλγόριθμοι είναι επιρρεπής σε κενές τιμές. Επίσης θετικό είναι η μη ύπαρξη διπλότυπων εγγραφών για την αξιοπιστία των αποτελεσμάτων της συσταδοποίησης.[23]

Στατιστικά Στοιχεία Δεδομένων

Έπειτα υπολογίζονται συνοπτικά στατιστικά στοιχεία για τις αριθμητικές μεταβλητές στο σύνολο δεδομένων, όπως ο μέσος όρος, η διάμεσος, η τυπική απόκλιση, οι ελάχιστες και οι μέγιστες τιμές. Αυτό μας δίνει μια επισκόπηση της κεντρικής τάσης, της εξάπλωσης και του εύρους των δεδομένων.

```
Summary statistics:
                AccX           AccY           AccZ          GyroX          GyroY          GyroZ
count  178360.000000  178360.000000  178360.000000  178360.000000  178360.000000  178360.000000
mean       -0.332545      -0.295320       9.300945      -0.114996      -0.081960       0.010386
std         1.693726       3.088376       1.814129       0.465687       0.520530       0.328347
min       -19.383743     -19.719864     -19.409454     -10.685318     -15.579224      -6.423645
25%        -0.876572      -2.355911       8.842133      -0.078018      -0.081863      -0.019760
50%        -0.389954      -1.113922       9.492455      -0.004517      -0.007187       0.003937
75%         0.298386       1.623840       9.936874       0.019073       0.022430       0.058716
max        19.689026      14.901489      19.663315      11.347015      10.550186       8.964951
```

Παράρτημα Δεδομένων 8: Στατιστικά στοιχεία δεδομένων αισθητήρων του Dataset

Από το παραπάνω παράρτημα παρατηρούμε ότι όλα τα χαρακτηριστικά έχουν το ίδιο σύνολο εγγραφών, από το οποίο αποτελείται και το dataset (178360 εγγραφές).

Ενώ όλα τα χαρακτηριστικά έχουν μέσο όρο κοντά στο μηδέν, το 'AccZ' έχει κοντά στο 9.30. Αυτό συμβαίνει γιατί ο άξονας Z του επιταχυνσιόμετρου είναι κάθετος στην επιτάχυνση της βαρύτητας (όταν το όχημα βρίσκεται σε επίπεδο έδαφος) και δείχνει μια τιμή ίση με την τιμή της επιτάχυνσης της βαρύτητας (περίπου 9.81 m/s^2 στην επιφάνεια της Γης). Τα υπόλοιπα χαρακτηριστικά του επιταχυνσιόμετρου λογικά έχουν μέσο όρο κοντά στο μηδέν, τιμή όταν το όχημα είναι σε αδράνεια (ακινητοποιημένο), αφού οι άξονες X και Y είναι παράλληλοι προς την επιτάχυνση της βαρύτητας. Αντίστοιχα όταν το όχημα είναι σε αδράνεια και δεν υπάρχει κίνηση ή περιστροφή,

τα δεδομένα από το γυροσκόπιο θα εμφανίζουν μηδενικές τιμές για τους τρεις άξονες, γεγονός που εξηγεί το μέσο όρο κοντά στο μηδέν.

Επίσης παρατηρούμε ότι για τους άξονες X και Y του επιταχυνσιόμετρου η ελάχιστη τιμή είναι κοντά στο -19.30 και η μέγιστη κοντά στο 19.70 και 14.90, ενώ ο μέσος όρος όπως είπαμε κοντά στο μηδέν, καθώς και η τυπική απόκλιση (std) είναι κοντά στο 1.70 και 3.00 αντίστοιχα. Επίσης το 75% δεν ξεπερνάει τις τιμές 0.29 και 1.62. Αυτές οι συγκρίσεις υποδεικνύουν μεγάλη απόκλιση και ίσως είναι ένδειξη ύπαρξης ακραίων σημείων σε αυτούς τους άξονες. Αν ισχύει αυτή η περίπτωση ίσως χρειαστεί η εφαρμογή μεθόδου ανίχνευσης και διαχείρισης ακραίων σημείων. Αντίστοιχα παρατηρούμε ότι για τους άξονες του γυροσκοπίου η ελάχιστη τιμή είναι κοντά στα -10.68, -15.58, -6.42 και η μέγιστη κοντά στα 11.35, 10.55, 8.96 ενώ ο μέσος όρος όπως είπαμε κοντά στο μηδέν, καθώς και η τυπική απόκλιση (std) είναι κοντά στο 0.50. Επίσης το 75% δεν ξεπερνάει τις τιμές 0.02 και 0.05. Αυτές οι συγκρίσεις επίσης υποδεικνύουν μεγάλη απόκλιση και ίσως είναι ένδειξη ύπαρξης ακραίων σημείων και στους άξονες του γυροσκοπίου.

Άλλη μια παρατήρηση είναι ότι η μέση απόκλιση του άξονα Y του επιταχυνσιόμετρου είναι η μεγαλύτερη από όλους τους άξονες και αν λάβουμε υπόψη ότι ο άξονας αυτός είναι η ένδειξη της επιτάχυνσης και της επιβράδυνσης του οχήματος, καταλαβαίνουμε ότι έχει τις περισσότερες εναλλαγές.

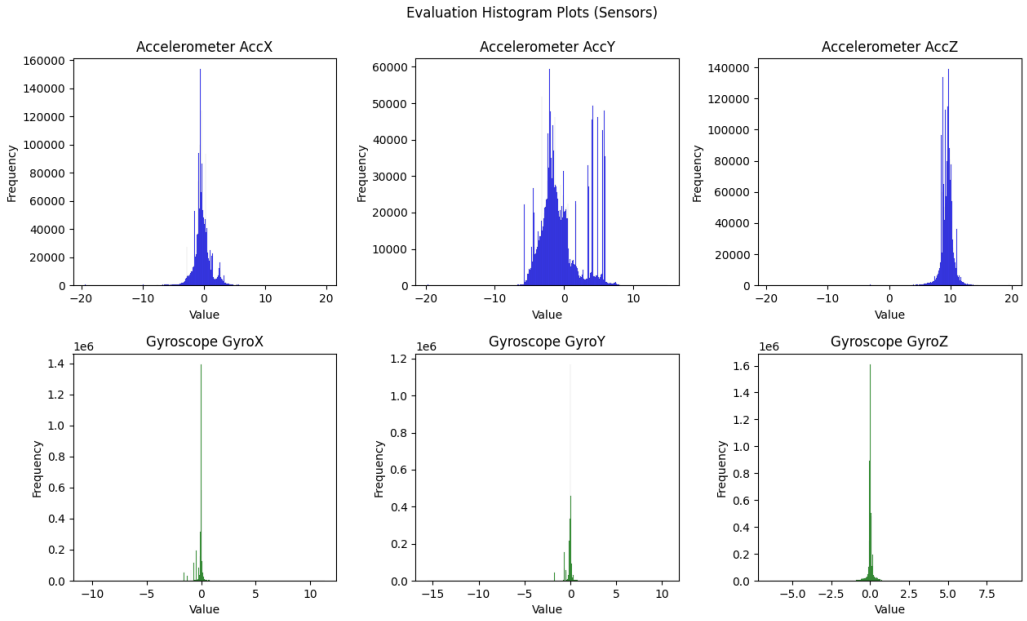
Τέλος, παρατηρούμε ότι οι τιμές του επιταχυνσιόμετρου και του γυροσκοπίου έχουν διαφορετική κλίμακα μέτρησης (του γυροσκοπίου η κλίμακα είναι μικρότερη), γεγονός που μας δίνει ένδειξη ότι ίσως να χρειαστεί η εφαρμογή κάποιας μεθόδου

κλιμάκωσης χαρακτηριστικών ή κανονικοποίησης δεδομένων έτσι ώστε να παραχθούν πιο αξιόπιστα αποτελέσματα.[23]

Οπτικοποιήσεις

Επόμενο βήμα είναι η δημιουργία οπτικοποιήσεων για την εξερεύνηση της διανομής και τις σχέσεις μεταξύ των μεταβλητών στο σύνολο δεδομένων. Οι συνήθεις τύποι οπτικοποιήσεων που χρησιμοποιούνται στο EDA περιλαμβάνουν διαγράμματα όπως histogram plot, Q-Q plot, Box Plot. Οι οπτικοποιήσεις βοηθούν στον εντοπισμό προτύπων, τάσεων, συστάδων και ακραίων στοιχείων στα δεδομένα[23]

Δημιουργήσαμε το *histogram plot* των δεδομένων των αισθητήρων, που αναπαριστά τη συχνότητα εμφάνισης των τιμών σε ένα σύνολο δεδομένων, χωρίζοντας το εύρος των τιμών σε διαφορετικά διαστήματα, γνωστά ως κατηγορίες ή διαστήματα. Ένα histogram plot μπορεί να παρέχει πολλές πληροφορίες για την κατανομή των δεδομένων και να προσφέρει σημαντικά συμπεράσματα. Μπορεί να προσδιοριστεί η μορφή κατανομής των δεδομένων, δηλαδή εάν οι τιμές των δεδομένων ακολουθούν κάποια συγκεκριμένη κατανομή, όπως η κανονική, η λογαριθμική κανονική, ή αν έχουν κάποια ανομοιογένεια. Μπορεί να γίνει εντοπισμός κενών ή ακραίων τιμών στα δεδομένα. Εάν υπάρχουν μεγάλες κενές περιοχές ή εξωφρενικές τιμές, αυτό μπορεί να επηρεάζει την αξιοπιστία των αναλύσεων. Μπορεί να γίνει εκτίμηση της τιμής της κεντρικής τάσης (μέση τιμή, διάμεσο, κλπ.), παρατηρώντας πού βρίσκονται οι περισσότερες τιμές, καθώς και αξιολόγηση της τιμής της διακύμανσης ή της απόκλισης από το μέσον των δεδομένων, βλέποντας το εύρος των τιμών και το πόσο συχνά εμφανίζονται σε σχέση με το κέντρο. Τέλος υπάρχει η δυνατότητα αναγνώρισης της μορφής της κατανομής, εάν είναι συμμετρική, αν η ουρά της κατανομής εκτείνεται περισσότερο προς τα αριστερά (αρνητική λοξότητα) ή προς τα δεξιά (θετική λοξότητα) και εάν υπάρχουν πολλαπλές κορυφές.



Σχήμα 34: Histogram Plot δεδομένων αισθητήρων

Για τον άξονα *AccX* του επιταχυνσιόμετρου, παρατηρούμε μια μεγάλη συγκέντρωση τιμών κοντά στο -1 που είναι λογικό όπως αναφέραμε πιο πάνω και ένα εύρος κατανομής των περισσότερων τιμών από το -7 έως το +4. Η ύπαρξη μερικών τιμών στο εύρος -7 έως -19 και +4 έως +19 υποδεικνύει ύπαρξη ακραίων τιμών που δεν είναι απαραιτήτως θόρυβος αλλά μπορεί να υποδεικνύουν ακραίες οδηγικές συμπεριφορές. Η κατανομή των δεδομένων ακολουθεί κατά βάση την κανονική κατανομή. Έχει μια μικρή αριστερή λοξότητα υποδεικνύοντας περισσότερες μικρότερες τιμές από την κεντρική τάση (-1) από ότι μεγαλύτερες. Η ύπαρξη μερικών κορυφών μικρότερης έκτασης από την κεντρική, υποδεικνύει την ύπαρξη δευτερευόντων ομάδων δεδομένων που ενδέχεται να αποτελούνται από υποομάδες ή κατηγορίες δεδομένων, με διαφορετικές ιδιότητες ή χαρακτηριστικά. Αυτές οι μικρότερες κορυφές μπορεί να υποδεικνύουν δηλαδή την ύπαρξη υποομάδων.

Για τον άξονα *AccY* του επιταχυνσιόμετρου, παρατηρούμε μια μεγάλη συγκέντρωση τιμών κοντά στο -2 που είναι λογικό όπως αναφέραμε πιο πάνω και ένα εύρος κατανομής των περισσότερων τιμών από το -6 έως το +8. Η ύπαρξη μερικών τιμών στο εύρος -6 έως -19 και +8 έως +19 υποδεικνύει ύπαρξη ακραίων τιμών, που δεν είναι απαραιτήτως θόρυβος, αλλά μπορεί να υποδεικνύουν ακραίες οδηγικές συμπεριφορές. Η κατανομή των δεδομένων ακολουθεί κατά βάση την κανονική κατανομή. Έχει μια μικρή δεξιά λοξότητα υποδεικνύοντας περισσότερες τιμές μεγαλύτερες από την κεντρική τάση (-2) από ότι μικρότερες. Η ύπαρξη μερικών

κορυφών μικρότερης έκτασης από την κεντρική αλλά πολύ πιο έντονες από του άξονα AccX, υποδεικνύει την ύπαρξη δευτερευόντων ομάδων δεδομένων που ενδέχεται να αποτελούνται από υποομάδες ή κατηγορίες δεδομένων με διαφορετικές ιδιότητες ή χαρακτηριστικά. Αυτές οι μικρότερες κορυφές μπορεί να υποδεικνύουν δηλαδή την ύπαρξη υποομάδων.

Για τον άξονα *AccZ* του επιταχυνσιόμετρου, παρατηρούμε μια μεγάλη συγκέντρωση τιμών κοντά στο +10 που είναι λογικό για τους λόγους που αναφέραμε πιο πάνω και ένα εύρος κατανομής των περισσότερων τιμών από το +5 έως το +13. Η ύπαρξη μερικών τιμών στο εύρος -19 έως +5 και +13 έως +19 υποδεικνύει ύπαρξη ακραίων τιμών που όμως εδώ φαίνεται ότι είναι σίγουρα θόρυβος αφού ο άξονας έχει σχέση με τη βαρύτητα και δεν μπορεί να έχει τόσο μεγάλη διαφοροποίηση στις τιμές. Η κατανομή των δεδομένων ακολουθεί κατά βάση την κανονική κατανομή. Έχει μια μικρή αριστερή λοξότητα υποδεικνύοντας περισσότερες τιμές μικρότερες της κεντρικής τάσης (+10) από ότι μεγαλύτερες. Η ύπαρξη μερικών κορυφών μικρότερης έκτασης από την κεντρική αλλά πολύ πιο έντονες από του άξονα AccX, υποδεικνύει την ύπαρξη δευτερευόντων ομάδων δεδομένων που ενδέχεται να αποτελούνται από υποομάδες ή κατηγορίες δεδομένων με διαφορετικές ιδιότητες ή χαρακτηριστικά. Αυτές οι μικρότερες κορυφές μπορεί να υποδεικνύουν δηλαδή την ύπαρξη υποομάδων.

Για τον άξονα *GyroX* του γυροσκοπίου, παρατηρούμε μια μεγάλη συγκέντρωση τιμών κοντά στο μηδέν που είναι λογικό όπως αναφέραμε πιο πάνω και ένα εύρος κατανομής των περισσότερων τιμών από το -1 έως το +1 με συντριπτικό αριθμό στο μηδέν. Η ύπαρξη μερικών τιμών στο εύρος -10 έως -1 και +1 έως +10 υποδεικνύει ύπαρξη ακραίων τιμών που εδώ μάλλον υποδηλώνουν θόρυβο μιας και ο συγκεκριμένος άξονας γυροσκοπίου δηλώνει κλίση οχήματος πάνω ή κάτω δηλαδή κλίση ανηφόρας κατηφόρας και δεν δικαιολογεί τόσο μεγάλη απόκλιση τιμών. Η κατανομή των δεδομένων ακολουθεί κατά βάση την κανονική κατανομή. Έχει μια μικρή αριστερή λοξότητα υποδεικνύοντας περισσότερες τιμές μικρότερες από την κεντρική τάση (0) από ότι μεγαλύτερες. Η ύπαρξη μερικών κορυφών μικρότερης έκτασης από την κεντρική, υποδεικνύει την ύπαρξη δευτερευόντων ομάδων δεδομένων που ενδέχεται να αποτελούνται από υποομάδες ή κατηγορίες δεδομένων με διαφορετικές ιδιότητες ή χαρακτηριστικά. Αυτές οι μικρότερες κορυφές μπορεί να υποδεικνύουν δηλαδή την ύπαρξη υποομάδων.

Για τον άξονα *GyroY* του γυροσκοπίου, παρατηρούμε μια μεγάλη συγκέντρωση τιμών κοντά στο μηδέν που είναι λογικό όπως αναφέραμε πιο πάνω και ένα εύρος κατανομής των περισσότερων τιμών από το -1 έως το +1 με συντριπτικό αριθμό στο μηδέν. Η ύπαρξη μερικών τιμών στο εύρος -15 έως -1 και +1 έως +10 υποδεικνύει ύπαρξη ακραίων τιμών που δεν υποδηλώνουν απαραίτητα θόρυβο. Η κατανομή των δεδομένων ακολουθεί κατά βάση την κανονική κατανομή. Δεν υπάρχει κάποια εμφανή λοξότητα. Η ύπαρξη μερικών κορυφών μικρότερης έκτασης από την κεντρική, υποδεικνύει την ύπαρξη δευτερευόντων ομάδων δεδομένων που ενδέχεται να αποτελούνται από υποομάδες ή κατηγορίες δεδομένων με διαφορετικές ιδιότητες ή χαρακτηριστικά. Αυτές οι μικρότερες κορυφές μπορεί να υποδεικνύουν δηλαδή την ύπαρξη υποομάδων.
Για τον άξονα *GyroZ* του γυροσκοπίου, παρατηρούμε μια μεγάλη συγκέντρωση τιμών κοντά στο μηδέν που είναι λογικό όπως αναφέραμε πιο πάνω και ένα εύρος κατανομής των περισσότερων τιμών από το -1.5 έως το +1.5 με συντριπτικό αριθμό στο μηδέν. Η ύπαρξη μερικών τιμών στο εύρος -5 έως -1.5 και +1.5 έως +7.5 υποδεικνύει ύπαρξη ακραίων τιμών που δεν υποδηλώνουν απαραίτητα θόρυβο. Η κατανομή των δεδομένων ακολουθεί κατά βάση την κανονική κατανομή. Δεν υπάρχει κάποια εμφανή λοξότητα. Η ύπαρξη μερικών κορυφών μικρότερης έκτασης από την κεντρική, υποδεικνύει την ύπαρξη δευτερευόντων ομάδων δεδομένων που ενδέχεται να αποτελούνται από υποομάδες ή κατηγορίες δεδομένων με διαφορετικές ιδιότητες ή χαρακτηριστικά. Αυτές οι μικρότερες κορυφές μπορεί να υποδεικνύουν δηλαδή την ύπαρξη υποομάδων.

Γενικό συμπέρασμα από αυτό το διάγραμμα, είναι ότι υπάρχει γενικά μια κανονική κατανομή, με ύπαρξη υποομάδων, που σημαίνει συσταδοποίηση με περισσότερες της μιας συστάδας, ύπαρξη ακραίων σημείων που υποδηλώνουν θόρυβο, αλλά κάποια σημεία από αυτά μπορεί να υποδηλώνουν και ακραίες οδηγικές συμπεριφορές. Οι άξονες που περιέχουν την περισσότερη πληροφορία, είναι οι AccY και GyroZ γεγονός καθόλου τυχαίο μιας και ο AccY μετράει την επιτάχυνση ή επιβράδυνση του οχήματος και ο GyroZ την γωνία στροφής του οχήματος αριστερά ή δεξιά. Δευτερεύοντες άξονες θεωρούνται οι AccX και GyroY με τον πρώτο να μετράει την ταχύτητα ολίσθησης του οχήματος προς τα αριστερά ή δεξιά και τον δεύτερο την κλίση του οχήματος κατά την πραγματοποίησης στροφής αριστερά ή δεξιά. Οι άξονες

AccZ και GyroX ίσως δεν βοηθήσουν στην εξαγωγή αξιόπιστων αποτελεσμάτων. Ειδικά ο AccZ που έχει σχέση με την ταχύτητα πτώσης ή ανύψωσης του οχήματος (σχετίζεται με την βαρύτητα) και με διαφορετική αφετηρία μέτρησης από τους άλλους μετρητές, θα συμβάλλει ελάχιστα. Ο GyroX μετράει την κλίση του οχήματος κατά την οδήγηση σε ανηφόρα ή κατηφόρα, οπότε θεωρείται και αυτός μικρής σημασίας για την αξιοποίηση του στην λήψη συμπερασμάτων για την εφαρμογή μας. Τέλος αναγνωρίζεται η ύπαρξη διαφορετικής κλιμάκωσης των χαρακτηριστικών του επιταχυνσιόμετρου με του γυροσκοπίου.

Διενεργώντας μέτρηση λοξότητας και κύρτωσης των δεδομένων, έγινε εξαγωγή των αποτελεσμάτων του παρακάτω παραρτήματος, που επιβεβαιώνουν τα προηγούμενα συμπεράσματα.

```
Skewness of each feature
(Skewness=0 for normal distribution):
  AccX    -1.541855
AccY     0.541078
AccZ    -5.221915
GyroX   -1.641719
GyroY    0.493993
GyroZ   -0.371632
dtype: float64
```

```
Kurtosis of each feature
(kurtosis=3 for normal distribution):
   AccX      18.683659
AccY      -0.055590
AccZ      45.946542
GyroX     60.191303
GyroY     86.713269
GyroZ    115.300483
dtype: float64
```

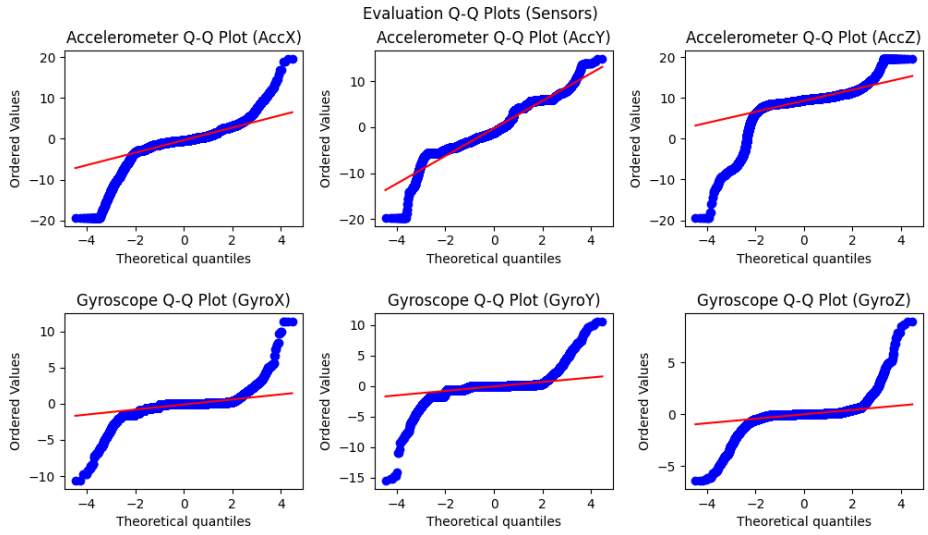
Παράρτημα Δεδομένων 9: Λοξότητα & Κύρτωση Δεδομένων

Η λοξότητα (skewness) στα δεδομένα αναφέρεται στην ασυμμετρία της κατανομής τους. Αποτελεί μια μέτρηση του βαθμού που μια κατανομή αποκλίνει από την κανονική κατανομή προς τη μία ή την άλλη κατεύθυνση. Συγκεκριμένα, αν η ουρά της κατανομής εκτείνεται περισσότερο προς τα αριστερά, τότε η λοξότητα είναι αρνητική. Αντίστοιχα, αν η ουρά της κατανομής εκτείνεται περισσότερο προς τα δεξιά, τότε η λοξότητα είναι θετική. Ένας μεγάλος βαθμός λοξότητας μπορεί να υποδεικνύει την παρουσία εκτροπών ή ανωμαλιών στα δεδομένα, ενώ μια κατανομή που είναι πιο κανονική θα έχει λοξότητα κοντά στο μηδέν. Από ότι παρατηρούμε οι άξονες AccZ και GyroX έχουν τον μεγαλύτερο βαθμό απόκλισης.

Η κύρτωση (kurtosis) αφορά το πόσο "στενή" ή "επίμονη" είναι η κορυφή μιας κατανομής και η ένταση των ουρών της. Αναφέρεται στην ποσότητα της συγκεντρωτικότητας των δεδομένων γύρω από τη μέση τιμή τους σε σχέση με μια τυπική κανονική κατανομή. Έτσι, μια κατανομή με υψηλή κύρτωση έχει μια πιο

στενή κορυφή και πιο βραχύτερες ουρές, ενώ μια κατανομή με χαμηλή κύρτωση έχει μια πιο εκτεταμένη κορυφή και πιο μακριές ουρές. Η κύρτωση συγκρίνεται με την κύρτωση μιας κανονικής κατανομής, που είναι 3. Αν η κύρτωση είναι μεγαλύτερη από 3, η κατανομή ονομάζεται υπέρ-κυρτωτική (leptokurtic), ενώ εάν είναι μικρότερη από 3, η κατανομή είναι πλατύκυρτη (platykurtic). Από ότι παρατηρούμε όλοι οι άξονες έχουν κύρτωση δεδομένων. Οι άξονες του γυροσκοπίου έχουν leptokurtic κύρτωση με τους μεγαλύτερους βαθμούς. Οι άξονες του επιταχυνσιόμετρου εκτός του AccY παρουσιάζουν κι αυτοί leptokurtic κύρτωση σε μικρότερο βαθμό από του γυροσκοπίου. Μόνο ο άξονας AccY παρουσιάζει platykurtic αποτέλεσμα των πολλών κορυφών οι οποίες είναι και αρκετά ψηλές.

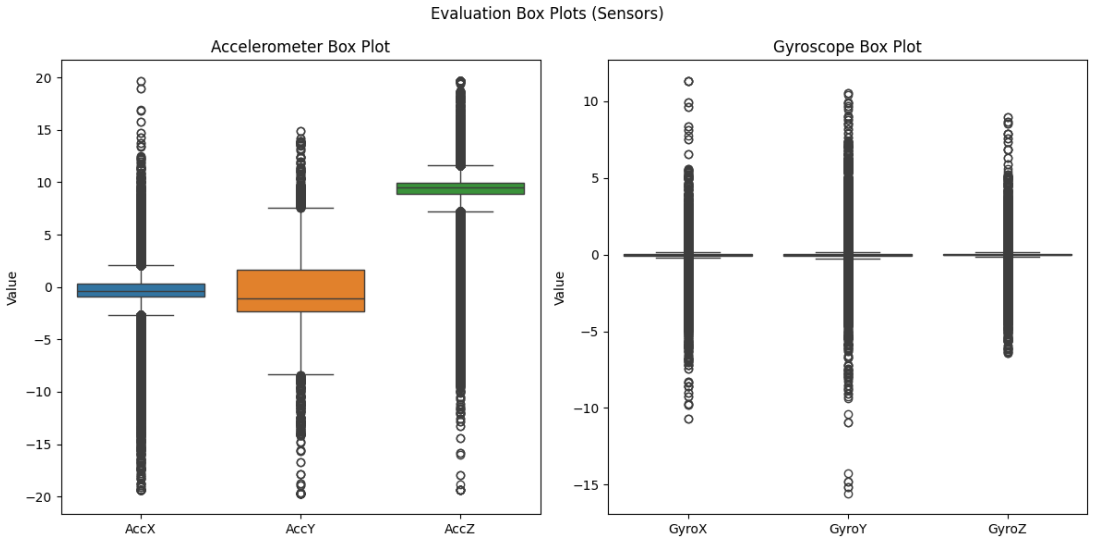
Στο επόμενο βήμα οπτικοποιούμε το *Q-Q (Quantile-Quantile) plot* που χρησιμοποιείται για να αξιολογήσει εάν δύο σύνολα δεδομένων προέρχονται από κατανομές που είναι παρόμοιες. Στην περίπτωσή μας χρησιμοποιείται για να ελέγξει εάν τα δεδομένα των αισθητήρων ακολουθούν μια κανονική κατανομή. Τα δεδομένα των αισθητήρων τοποθετούνται στον άξονα x, ενώ οι αντίστοιχες κανονικές τιμές που αναμένουμε να βρούμε αν η κατανομή των δεδομένων είναι κανονική τοποθετούνται στον άξονα y. Αν οι δεδομένες τιμές προέρχονται από μια κανονική κατανομή, τότε τα σημεία του Q-Q plot θα ακολουθούν μια γραμμική τάση. Αν υπάρχουν αποκλίσεις από την γραμμική τάση, αυτό υποδεικνύει ότι τα δεδομένα δεν ακολουθούν μια κανονική κατανομή.



Σχήμα 35: Q-Q Plot των δεδομένων των αισθητήρων

Παρατηρούμε ότι όλοι οι άξονες δεν ακολουθούν επακριβώς μια κανονική κατανομή με αρκετές αποκλίσεις. Την μεγαλύτερη συνοχή με την κανονική κατανομή δείχνει να έχει ο AccY. Την χειρότερη συνοχή έχει ο AccZ. Οι άξονες του γυροσκοπίου έχουν παρόμοια κατανομή ακολουθώντας την κανονική στο εύρος δεδομένων -2 έως +2. Σε συνδυασμό με τα προηγούμενα συμπεράσματα του histogram plot η κλιμάκωση των σημείων της κανονικής κατανομής σχεδόν ταυτίζονται και αναδεικνύεται ότι τα σημεία εκτός κανονικής κατανομής είναι ακραία σημεία.

Επόμενο βήμα είναι η οπτικοποίηση του *Box plot* που χρησιμοποιείται για να παρουσιάσει τη διανομή των δεδομένων και την πιθανή παρουσία εκτός του κύριου συνόλου τα ακραία σημεία (outliers). Παρέχει μια εικόνα της κεντρικής τάσης, της διασποράς, και της μορφής της κατανομής των δεδομένων. Απεικονίζει τα πέντε κύρια στατιστικά μέτρα των δεδομένων των αισθητήρων: Την ελάχιστη τιμή (minimum), το πρώτο τεταρτημόριο (1st quartile), τη μέση τιμή ή το δεύτερο τεταρτημόριο, (median), το τρίτο τεταρτημόριο (3rd quartile) και τη μέγιστη τιμή (maximum). Απεικονίζει τις τιμές εκτός του κύριου συνόλου δεδομένων ως ξεχωριστά ακραία σημεία (outliers) εκτός του κουτιού (box). Είναι ένα ιδιαίτερα χρήσιμο εργαλείο για την ανάλυση της κατανομής των δεδομένων και την εντοπισμό τυχόν ανωμαλιών.



Σχήμα 36: Box plot δεδομένων αισθητήρων

Παρατηρούμε ότι για τα δεδομένα του γυροσκοπίου το εύρος τιμών μεταξύ της ελάχιστης και μέγιστης τιμής είναι πολύ περιορισμένο, λίγο πάνω-κάτω από το μηδέν,

κάτι που δεν κάνει εντύπωση λαμβάνοντας υπόψη τα συμπεράσματα από τα προηγούμενα διαγράμματα και στατιστικά στοιχεία. Φαίνεται πολύ ξεκάθαρα ότι τα ακραία σημεία αποτελούν μεγάλο μέρος των δεδομένων του γυροσκοπίου.
Τα ακραία σημεία κάνουν αισθητή την παρουσία τους και στα δεδομένα του επιταχυνσιόμετρου. Η διαφορά είναι ότι το εύρος των κανονικών δεδομένων μεταξύ ελαχίστου και μεγίστου είναι αρκετά πιο διευρυμένο, με μεγαλύτερο εύρος πληροφορίας στο AccY.
Η μέση τιμή, τα όρια, η τάση της πληροφορίας και η κλιμάκωση των δεδομένων σε αυτό το διάγραμμα, συμφωνεί με τα προηγούμενα διαγράμματα.[23]

Υλοποίηση Εφαρμογής

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε ο αλγόριθμος της μάθησης χωρίς επίβλεψη για συσταδοποίηση K-means. Πριν την εκτέλεση του αλγόριθμου K-means εφαρμόζεται μέθοδος μείωσης των διαστάσεων των χαρακτηριστικών των δεδομένων σε δύο (PCA). Αυτή η εφαρμογή θα βοηθήσει στην μείωση χρήσης υπολογιστικών πόρων καθώς και στην καλύτερη οπτικοποίηση των συσταδοποιήσεων. Η υλοποίηση διαχωρίστηκε σε δύο βασικά στάδια:

- Ανίχνευση καλύτερης ποιότητας συσταδοποίησης βάση σεναρίων
- Εκτέλεση σεναρίου οπτικοποίησης συσταδοποίησης

Ανίχνευση καλύτερης ποιότητας συσταδοποίησης βάση σεναρίων

Στο πρώτο στάδιο συγκαταλέγεται η εκτέλεση μιας ομάδας προκαθορισμένων σεναρίων συνδυασμού διαδικασιών προ-επεξεργασίας και μετασχηματισμού, όπως ανίχνευση ακραίων σημείων (outliers detection), κανονικοποίηση δεδομένων (normalization) και κλιμάκωσης χαρακτηριστικών των δεδομένων (features scaling). Για το κάθε σενάριο εκτελούνται μετρικές αξιολόγησης της ποιότητας της συσταδοποίησης για την ανίχνευση του βέλτιστου αριθμού συστάδων εφαρμογής στον αλγόριθμο K-means. Αυτές οι μετρικές όπως προαναφέρθηκαν στην ανάλυση της εφαρμογής είναι οι inertia, silhouette score, Calinski-Harabasz index και Davies-

Bouldin index. Στόχος αυτού του σταδίου είναι να βρεθεί το σενάριο που αποδίδει καλύτερα όσον αφορά στην ποιότητα της συσταδοποίησης καθώς και ο βέλτιστος αριθμός συστάδων του σεναρίου αυτού.

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες της python:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import normalize
from scipy.stats.mstats import winsorize
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
import matplotlib.pyplot as plt
import zipfile
import sys
import os
from pathlib import Path
```

Παράρτημα Κώδικα 12: Βιβλιοθήκες python αρχείου DriverPredictSenarios_(Kmeans).py

Η υλοποίηση του σταδίου αυτού αποτελείται από τα εξής βήματα:

1. Φόρτωση των δεδομένων από το συμπιεσμένο αρχείο με το τελικό dataset που δημιουργήθηκε στο στάδιο της προ-επεξεργασίας. Το τελικό dataset είναι σε μορφή συμπιεσμένου αρχείου zip με όνομα finalData.csv.zip. Μέσα σε αυτό βρίσκεται το finalData.csv, το οποίο φορτώνεται στην εφαρμογή μας μέσω του κώδικα:

```
#---  Read Data from Zip CSV file  ------------------------------------------------------
file=Path("c:/datasets/finalData.csv.zip")
if not os.path.isfile(file):
    print("Dataset: C:/Datasets/finalData.csv.zip not exists!")
    sys.exit()
zf = zipfile.ZipFile('C:/Datasets/finalData.csv.zip')
df = pd.read_csv(zf.open('finalData.csv'))
#-------------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 13: Φόρτωση Dataset

2. Φόρτωση των δεδομένων των σεναρίων. Αναλόγως της σειράς εκτέλεσης που επιλέγεται από τον χρήστη φορτώνονται τα δεδομένα που περιέχονται στα αρχεία με όνομα executionSenarios.csv ή executionSenarios2.csv μέσω του

κώδικα:

```
#---  Read Senarios Data from CSV file  ------------------------------------------------
senarioAnswer = int(input("Choose Senarios Group (1/2):\n"+
                          "1: Normalization-Features Scaling\n"+
                          "2: Features Scaling-Normalization\n"+
                          "Enter Choise: "))
if senarioAnswer==1:
    file=Path("c:/datasets/executionSenarios.csv")
    if not os.path.isfile(file):
        print("Dataset: C:/Datasets/executionSenarios.csv not exists!")
        sys.exit()
    dfSenarios = pd.read_csv('C:\datasets\executionSenarios.csv')
elif senarioAnswer==2:
    file=Path("c:/datasets/executionSenarios2.csv")
    if not os.path.isfile(file):
        print("Dataset: C:/Datasets/executionSenarios2.csv not exists!")
        sys.exit()
    dfSenarios = pd.read_csv('C:\datasets\executionSenarios2.csv')
else:
    print("Wrong Choise!")
    sys.exit()
#-------------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 14: Φόρτωση Δεδομένων Σεναρίων

3. Διαδικασία τυχόν φιλτραρίσματος των δεδομένων για μεμονωμένη επεξεργασία. Δίνεται η επιλογή στον χρήστη αν θα χρησιμοποιήσει μέρος του dataset ή όλα τα δεδομένα μέσω του κώδικα:

```
#---  Filtering  ----------------------------------------------------------------------
dfFiltered, row_count = filterData(df)
print(f"The Filtered Data have {row_count} Rows!")
if row_count <= 0:
    sys.exit()
df_scaled = dfFiltered[features]
# Find the more appropriate number of samples for silhouette score
score_sample = round(row_count * 5 / 100)
#-------------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 15: Φιλτράρισμα Δεδομένων

Επίσης γίνεται έλεγχος αν υπάρχουν εγγραφές και αν όχι η εφαρμογή τερματίζει.

Το φιλτράρισμα των δεδομένων πραγματοποιείται μέσω της συνάρτησης filterData της οποίας ο κώδικας είναι:

```
def filterData(df):
    flt_s = input("\nFilter application (0/1/2/3)\n"+
                                "0: None\n"+
                                "1: Driver\n"+
                                "2: DayPart\n"+
                                "3: Driver & DayPart\n"+
                                "Enter choice: ")
    print("you chose: " + flt_s + "\n")
    flt = int(flt_s)

    if flt==0:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle.append('All')
    elif flt==1:
        print("\nDataset will be filtered for some driver!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        dfFiltered = df.loc[df['Driver'] == flt1]
        plotTitle.append('Driver'+flt1_s)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['DayPart']))
            print('\nDriver'+flt1_s+' made ' + str(n) + ' roots\n')
    elif flt==2:
        print("\nDataset will be filtered for some part of the Day!\n")
        flt1_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt1_s + "\n")
        flt1 = float(flt1_s)
        dfFiltered = df.loc[df['DayPart'] == flt1]
        if flt1==0.25:
            title1="Morning"
        elif flt1==0.5:
            title1="Afternoon"
        elif flt1==0.75:
            title1="Evening"
        elif flt1==1.0:
            title1="Night"
        else:
            title1="All"
        plotTitle.append('Part of Day '+title1)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['Driver']))
            print('\nThe Part of Day '+flt1_s+' made by ' + str(n) + ' Drivers\n')
    elif flt==3:
        print("\nDataset will be filtered for a compination of Driver AND part of the Day!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        title1 = 'Driver'+flt1_s
        flt2_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt2_s + "\n")
        flt2 = float(flt2_s)
        dfFiltered = df[(df.Driver == flt1) & (df.DayPart == flt2)]
        if flt2==0.25:
            title2="Morning"
        elif flt2==0.5:
            title2="Afternoon"
        elif flt2==0.75:
            title2="Evening"
        elif flt2==1.0:
            title2="Night"
        else:
            title2="All"
        plotTitle.append(title1 + ' and ' + title2)
    else:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle.append('All')

    row_count = len(dfFiltered)

    return dfFiltered, row_count
```

Παράρτημα Κώδικα 16: Συνάρτηση Φιλτραρίσματος Δεδομένων

Μέσω αυτής της συνάρτησης ο χρήστης έχει την δυνατότητα να επιλέξει να έχει στην διάθεσή του όλα τα δεδομένα, τα δεδομένα ενός συγκεκριμένου

οδηγού, ενός συγκεκριμένου μέρους της ημέρας ή συνδυασμό οδηγού – μέρους της ημέρας.

4. Πραγματοποιείται επιλογή σεναρίου
5. Βάση σεναρίου πραγματοποιείται ή όχι η εφαρμογή μεθόδου ανίχνευσης και διαχείρισης ακραίων σημείων στα δεδομένα.

```
df_scaled = outliersDetection(features, winsorizeVal, df_scaled)
if winsorizeVal==1:
    print("Senario " + str(ind) + ": You performed Outliers Detection with Winsorize!")
```

Παράρτημα Κώδικα 17: Εφαρμογή μεθόδου Διαχείρισης Ακραίων Σημείων

Η διαδικασία πραγματοποιείται μέσω της συνάρτησης ouliersDetection:

```
def outliersDetection(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1:
        # Apply Winsorizer to data
        df1 = df
        for column in features:
            df1.loc[:, column] = winsorize(df1[column], limits=[0.01, 0.01])
        plotTitle.append('Winsorizer')
    return df1
```

Παράρτημα Κώδικα 18: Συνάρτηση μεθόδου Διαχείρισης Ακραίων Σημείων

6. Βάση σεναρίου πραγματοποιείται ή όχι η εφαρμογή κάποιας από τις μεθόδους κανονικοποίησης των δεδομένων μας (L1, L2, Max).

```
#---  Normalization Methods  -------------------------------------------------------
if l1Val==1:
    answer=1
    print("Senario " + str(ind) + ": You performed Normalization with Normalize (L1)!")
elif l2Val==1:
    answer=2
    print("Senario " + str(ind) + ": You performed Normalization with Normalize (L2)!")
elif maxVal==1:
    answer=3
    print("Senario " + str(ind) + ": You performed Normalization with Normalize (Max)!")
else:
    answer=0
df_scaled = normalization(features, answer, df_scaled)
#---------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 19: Κανονικοποίηση Δεδομένων

Η διαδικασία πραγματοποιείται μέσω της συνάρτησης normalization:

```
def normalization(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1 or answer==2 or answer==3:
        if answer==1: norm='l1'
        if answer==2: norm='l2'
        if answer==3: norm='max'
        # Apply Normalization to data
        df1 = normalize(df, norm=norm)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Normalization ' + norm)
    return df1
```

Παράρτημα Κώδικα 20: Συνάρτηση Κανονικοποίησης Δεδομένων

7. Βάση σεναρίου πραγματοποιείται ή όχι η εφαρμογή κάποιας από τις μεθόδους κλιμάκωσης χαρακτηριστικών των δεδομένων μας έτσι ώστε ανόμοιες μετρήσεις από διαφορετικούς αισθητήρες να κλιμακωθούν σε ενιαία κλίμακα (StandardScaler, MinMaxScaler, RobustScaler).

```
#---  Features Scaling Methods  ----------------------------------------------------
if standardScalerVal==1:
    answer=1
    print("Senario " + str(ind) + ": You performed Features Scaling (StandardScaler)!")
elif minMaxScalerVal==1:
    answer=2
    print("Senario " + str(ind) + ": You performed Features Scaling (MinMaxScaler)!")
elif robustScalerVal==1:
    answer=3
    print("Senario " + str(ind) + ": You performed Features Scaling (RobustScaler)!")
else:
    answer=0
df_scaled = featuresScaling(features, answer, df_scaled)
#-------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 21: Κλιμάκωση Χαρακτηριστικών

Η διαδικασία πραγματοποιείται μέσω της συνάρτησης featuresScaling:

```
def featuresScaling(features, preProp, df):
    if preProp==0:
        df1 = df
    elif preProp==1:
        # Standardize the features for clustering
        scaler = StandardScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Standardize')
    elif preProp==2:
        # Apply MinMaxScaler to data
        scaler = MinMaxScaler(feature_range=(-1, 1))
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('MinMaxScaler')
    elif preProp==3:
        # Apply RobustScaler to data
        scaler = RobustScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('RobustScaler')
    elif preProp==4:
        # Apply QuantileTransformer to data
        scaler = QuantileTransformer( n_quantiles=10000, random_state=45)
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('QuantileTransformer')
    return df1
```

Παράρτημα Κώδικα 22: Συνάρτηση Κλιμάκωσης Χαρακτηριστικών

8. Εφαρμογή μεθόδου μείωσης διαστάσεων σε δύο έτσι ώστε να είναι πιο αποδοτική η οπτική αξιολόγηση, καθώς και η πολυπλοκότητα επεξεργασίας.

```
#---  PCA Method  ----------------------
dfFinal = methodPCA(df_scaled, 1)
#---------------------------------------
```

Παράρτημα Κώδικα 23: Μείωση Διαστάσεων (PCA)

Η διαδικασία πραγματοποιείται μέσω της συνάρτησης methodPCA:

```
def methodPCA(df_scaled, num):
    # Create PCA object for Axis X
    pca_X = PCA(n_components=num)
    pcaAcc = pca_X.fit_transform(df_scaled[['AccX', 'AccY', 'AccZ']])

    # Create PCA object for Axis Y
    pca_Y = PCA(n_components=num)
    pcaGyro = pca_Y.fit_transform(df_scaled[['GyroX', 'GyroY', 'GyroZ']])

    # Concatenate PCA results
    dfFinal = pd.DataFrame({
        'Acc': pcaAcc.flatten(),
        'Gyro': pcaGyro.flatten()
    })
    plotTitle.append('PCA')

    return dfFinal
```

Παράρτημα Κώδικα 24: Συνάρτηση Μείωσης Διαστάσεων (PCA)

Η μείωση των έξι διαστάσεων σε δύο πραγματοποιείται μειώνοντας πρώτα τις τρεις διαστάσεις που αποτελούν τα χαρακτηριστικά του αισθητήρα Accelerator (AccX, AccY, AccZ) σε μία διάσταση (Acc) και κατόπιν τις τρεις διαστάσεις που αποτελούν τα χαρακτηριστικά του αισθητήρα Gyroscope (GyroX, GyroY, GyroZ) σε μία διάσταση (Gyro). Με αυτό τον τρόπο έχουμε συμπτύξει τα δεδομένα του Accelerator τα οποία αφορούν αυτόν και μόνο τον αισθητήρα σε μια διάσταση, και τα δεδομένα του Gyroscope σε μια άλλη. Αυτό θα μας βοηθήσει να έχουμε καλύτερη οπτική απεικόνιση των δεδομένων μας χρησιμοποιώντας ένα διάγραμμα όπως θα δούμε παρακάτω, καθώς και επιτυγχάνουμε ταχύτερη επεξεργασία των αλγορίθμων

9. Επιλογή αριθμού συστάδων για εφαρμογή των δεδομένων του σεναρίου στον αλγόριθμο K-means και υπολογισμού μετρικών αξιολόγησης ποιότητας συσταδοποίησης.
10. Εφαρμογή του αλγορίθμου μηχανικής μάθησης χωρίς επίβλεψη για συσταδοποίηση (clustering) K-means για τον συγκεκριμένο αριθμό συστάδων.

```
cluster_range = range(2, 11)

for k in cluster_range:
    kmeans = KMeans(n_clusters=k, init="k-means++", random_state=45)
    cluster_labels = kmeans.fit_predict(dfFinal)
```

Παράρτημα Κώδικα 25: Εφαρμογή K-means

11. Εκτέλεση υπολογισμού και καταχώρησης των τεσσάρων μετρικών αξιολόγησης ποιότητας συσταδοποίησης για τα αποτελέσματα του

```
labels_per_cluster[k] = cluster_labels
inertia_per_cluster[k] = kmeans.inertia_
cluster_centers_per_cluster[k] = kmeans.cluster_centers_
inertia.append(inertia_per_cluster[k])
silhouette_scores_per_cluster[k] = silhouette_score(dfFinal, cluster_labels,
                                                    sample_size=score_sample, random_state=45)
silhouette_scores.append(silhouette_scores_per_cluster[k])
calinski_harabasz_scores_per_cluster[k] = calinski_harabasz_score(dfFinal, cluster_labels)
calinski_harabasz_scores.append(calinski_harabasz_scores_per_cluster[k])
davies_bouldin_scores_per_cluster[k] = davies_bouldin_score(dfFinal, cluster_labels)
davies_bouldin_scores.append(davies_bouldin_scores_per_cluster[k])
```

Παράρτημα Κώδικα 26: Υπολογισμός & Καταχώρηση Μετρικών Αξιολόγησης Ποιότητας Συσταδοποίησης

12. Εκτέλεση βημάτων 9-11 για το εύρος αριθμού συστάδων 2-11
13. Οπτικοποίηση των τεσσάρων μετρικών για το εύρος αριθμού συστάδων 2-11

```
# Plot evaluation metrics
fig, ax = plt.subplots(2, 2, figsize=(12, 10))

# Inertia
ax[0, 0].plot(cluster_range, inertia, marker='o', color='red')
ax[0, 0].set_title('Elbow Method', color='red')
ax[0, 0].set_xlabel('Number of Clusters')
ax[0, 0].set_ylabel('Inertia')

# Silhouette Score
ax[0, 1].plot(cluster_range, silhouette_scores, marker='o', color='green')
ax[0, 1].set_title('Silhouette Score', color='green')
ax[0, 1].set_xlabel('Number of Clusters')
ax[0, 1].set_ylabel('Silhouette Score')

# Davies-Bouldin Index
ax[1, 0].plot(cluster_range, davies_bouldin_scores, marker='o', color='purple')
ax[1, 0].set_title('Davies-Bouldin Index', color='purple')
ax[1, 0].set_xlabel('Number of Clusters')
ax[1, 0].set_ylabel('Davies-Bouldin Index')

# Calinski-Harabasz Index
ax[1, 1].plot(cluster_range, calinski_harabasz_scores, marker='o', color='blue')
ax[1, 1].set_title('Calinski-Harabasz Index', color='blue')
ax[1, 1].set_xlabel('Number of Clusters')
ax[1, 1].set_ylabel('Calinski-Harabasz Index')


pltTitle = createPlotTitle(plotTitle)
pltTitle = 'Evaluation (' + pltTitle + ')'
plt.suptitle(pltTitle)
plt.savefig("c:/datasets/plots/evalNumClusters/" + pltTitle + ".png")
plt.subplots_adjust(top = 0.9, bottom = 0.1, hspace=0.5)
#plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=1.0)
plt.show()
```

Παράρτημα Κώδικα 27: Οπτικοποίηση των Μετρικών για εύρος συστάδων 2-10

14. Επιλογή του βέλτιστου αριθμού συστάδων από τον χρήστη βάση του βήματος 13.

```
#---  Based on the evaluation, set the optimal number of clusters  ------------------
k_optimal = int(input("Senario " + str(ind) + ": Enter the number of optimal Clusters:"))
#-------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 28: Επιλογή Βέλτιστου Αριθμού Συστάδων

15. Καταχώρηση των τιμών των μετρικών για τον βέλτιστο αριθμού συστάδων του σεναρίου καταχωρούνται σε dataframe.

```
inertiaVal      = round(inertia_per_cluster[k_optimal],3)
silhouetteVal   = round(silhouette_scores_per_cluster[k_optimal],3)
calinskiVal     = round(calinski_harabasz_scores_per_cluster[k_optimal],3)
daviesVal       = round(davies_bouldin_scores_per_cluster[k_optimal],3)

pltTitle = createPlotTitle(plotTitle)
if senarioAnswer==1:
    sign='s'
elif senarioAnswer==2:
    sign='r'

record = [sign+senario,
          pltTitle,
          k_optimal,
          inertiaVal,
          silhouetteVal,
          calinskiVal,
          daviesVal
          ]
dfMetrics.loc[len(dfMetrics)] = record
```

Παράρτημα Κώδικα 29: Καταχώρηση Μετρικών Βέλτιστου Αριθμού Συστάδων Σεναρίου

16. Εκτέλεση των βημάτων 4 – 15 για το σύνολο των σεναρίων
17. Υπολογισμός και καταχώρηση του τελικού σκορ του κάθε σεναρίου

```
# Normalize scores values in scale
minVal = dfMetrics['inertia'].min()
maxVal = dfMetrics['inertia'].max()
inertiaNorm = (dfMetrics['inertia'] - minVal) / (maxVal - minVal)
minVal = dfMetrics['silhouette'].min()
maxVal = dfMetrics['silhouette'].max()
silhouetteNorm = (dfMetrics['silhouette'] - minVal) / (maxVal - minVal)
minVal = dfMetrics['calinskiHarabasz'].min()
maxVal = dfMetrics['calinskiHarabasz'].max()
calinskiHarabaszNorm = (dfMetrics['calinskiHarabasz'] - minVal) / (maxVal - minVal)
minVal = dfMetrics['davisBouldin'].min()
maxVal = dfMetrics['davisBouldin'].max()
davisBouldinNorm = (dfMetrics['davisBouldin'] - minVal) / (maxVal - minVal)


# Calculate total score for each cluster number
dfMetrics['totalScore'] = ((-1.0) * inertiaNorm) + silhouetteNorm + calinskiHarabaszNorm + ((-1.0) * davisBouldinNorm)
```

Παράρτημα Κώδικα 30: Υπολογισμός & Καταχώρηση Τελικού Σκορ κάθε Σεναρίου

18. Οπτικοποίηση των τελικών σκορ του κάθε σεναρίου

```
#==  Create Metrics Plot  ================================================
# Get unique clusters
clusters = dfMetrics['clusters'].unique()
nums = len(clusters)
# Generate colors for clusters
colors = plt.cm.tab10(np.linspace(0, 1, nums))

plt.figure(figsize=(14, 7))
# Plot bars for each cluster
for i, cluster in enumerate(clusters):
    clusterNums = dfMetrics[dfMetrics['clusters'] == cluster]
    plt.bar(clusterNums['senario'], clusterNums['totalScore'], color=colors[i], label=f'Clusters {cluster}')
plt.title('Metrics Scores (Inertia, Silhouette, Calinski Harabasz, Davis Bouldin) Plot')
plt.xlabel('Senario')
plt.ylabel('Total Score')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='dotted')
plt.legend()
plt.tight_layout()
plt.savefig("c:/datasets/plots/evalNumClusters/clustersMetrics.png")
plt.show()
#=========================================================================
```

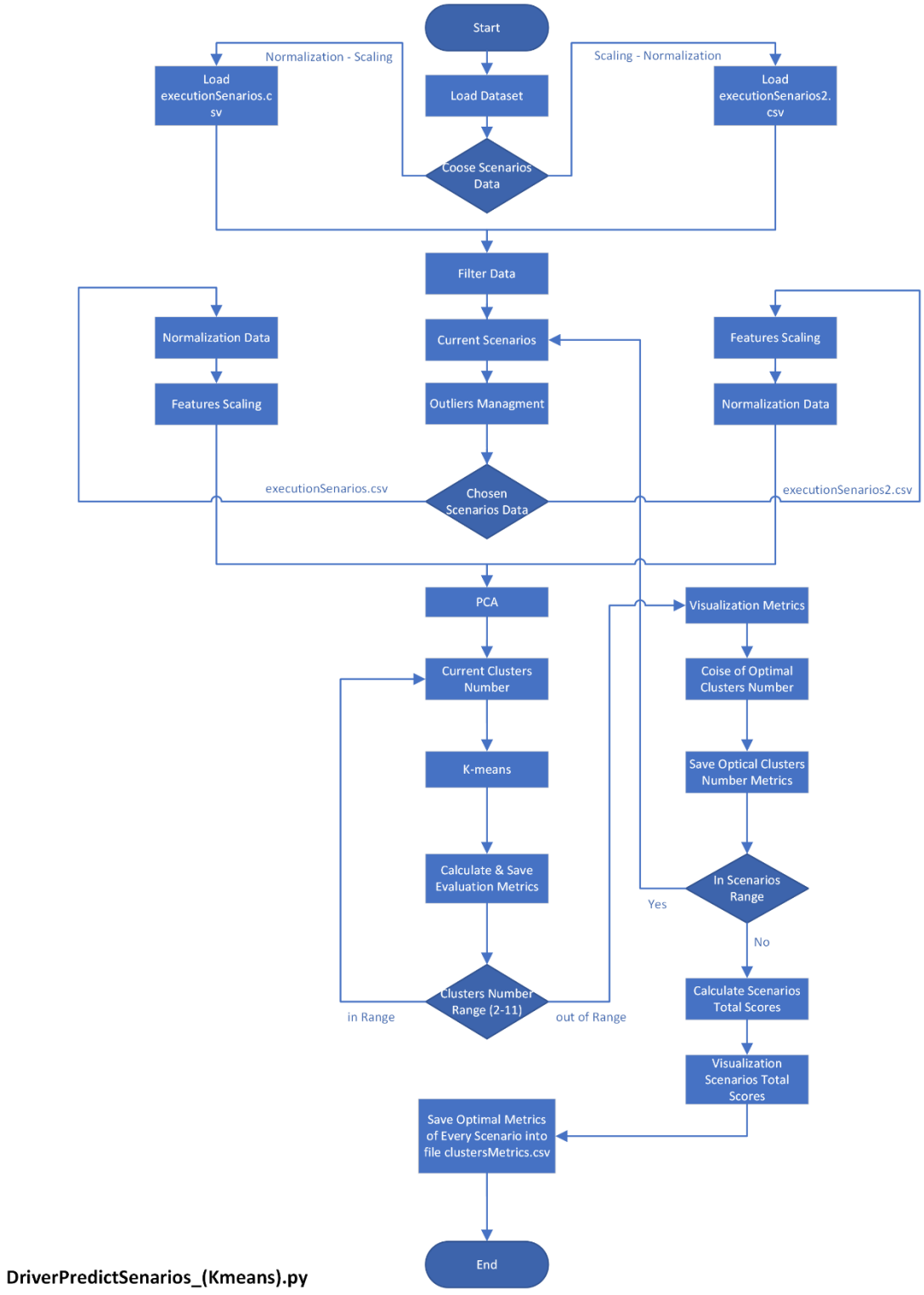
Παράρτημα Κώδικα 31: Οπτικοποίηση Τελικών Σκορ κάθε Σεναρίου

19. Καταχώρηση των μετρικών βέλτιστου αριθμού συστάδων για το κάθε σενάριο σε αρχείο με όνομα clustersMetrics.csv

```
print("Save Metrics Data locally as clustersMetrics.csv...\n")
dfMetrics.to_csv('C:/datasets/clustersMetrics.csv', index=False)
```

Παράρτημα Κώδικα 32: Καταχώρηση Μετρικών Βέλτιστου Αριθμού Συστάδων για κάθε Σενάριο

Στο παρακάτω διάγραμμα ροής, γίνεται παρουσίαση των παραπάνω βημάτων εκτέλεσης:



Σχήμα 37: Διάγραμμα Ροής DriverPredictSenarios_(Kmeans).py

Ο κώδικας του αρχείου που εκτελεί την διαδικασία ανίχνευσης καλύτερης ποιότητας συσταδοποίησης βάση σεναρίων, βρίσκεται στην τοποθεσία:

https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link με όνομα ***DriverPredictSenarios_(Kmeans).py***

Εκτέλεση σεναρίου οπτικοποίησης συσταδοποίησης

Σε αυτό το στάδιο γίνεται εκτέλεση του καλύτερου σεναρίου που ανιχνεύτηκε στο προηγούμενο στάδιο και γενικά οποιουδήποτε σεναρίου. Στόχος αυτού του σταδίου είναι η οπτικοποίηση της συσταδοποίησης αποτέλεσμα τυχόν συνδυασμού μετασχηματισμού δεδομένων και της εφαρμογής του K-means, καθώς και πληροφοριών που αφορούν τις συστάδες και τα κεντροειδή τους. Μπορεί να επιτελεστεί φιλτράρισμα δεδομένων κάποιου συγκεκριμένου οδηγού, μέρος της ημέρας ή συνδυασμού αυτών, έτσι ώστε να ληφθούν μεμονωμένα συμπεράσματα για τα δεδομένα αυτά.

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες της python:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import normalize
from scipy.stats.mstats import winsorize
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
import matplotlib.pyplot as plt
import zipfile
import seaborn as sns
import sys
import dataframe_image as dfi
from PIL import Image, ImageDraw, ImageFont
```

Παράρτημα Κώδικα 33: Βιβλιοθήκες python για εφαρμογή K-means

Η υλοποίηση της εφαρμογής πραγματοποιήθηκε στα εξής βήματα:

1. Φόρτωση των δεδομένων από το συμπιεσμένο αρχείο με το dataset που δημιουργήθηκε στο στάδιο της προ-επεξεργασίας με όνομα finalData.csv.zip. Μέσα σε αυτό βρίσκεται το finalData.csv, το οποίο φορτώνεται στην εφαρμογή μας μέσω του κώδικα:

```
#---  Read Data from Zip CSV file  ------------------------------------------------------
zf = zipfile.ZipFile('C:/Datasets/finalData.csv.zip')
df = pd.read_csv(zf.open('finalData.csv'))
#-----------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 34: Φόρτωση συμπιεσμένου αρχείου με το Dataset

2. Επιλογή συμμετοχής χαρακτηριστικών των αισθητήρων.

```
#---  Selection of Sensors Features  ------------------
answer = input("\nChoose Accelerator Features (XYZ): ")
answer=answer.upper()
fAcc = []
if all(char in 'XYZ' for char in answer):
    for axis in answer:
        fAcc.append('Acc'+axis)
print("you chose: " + str(fAcc) + "\n")

answer = input("\nChoose Gyroscope Features (XYZ): ")
answer=answer.upper()
fGyro = []
if all(char in 'XYZ' for char in answer):
    for axis in answer:
        fGyro.append('Gyro'+axis)
print("you chose: " + str(fGyro) + "\n")
featuresSensors=fAcc+fGyro
fTime=['Millsecs']
answer = input("\nHow will the two dimensions be separated? (1/2)\n"+
               "1: Accelerator - Gyroscope\n"+
               "2: Time - Sensors\n"+
               "Enter Choise: ")
features=[]
featuresX=[]
featuresY=[]
if all(char in '12' for char in answer):
    if answer=='2':
        features=fTime+featuresSensors
        featuresX=fTime
        featuresY=featuresSensors
        titleAxisX='Time'
        titleAxisY='Sensors ['
        for item in featuresSensors:
            titleAxisY=titleAxisY+item+', '
        titleAxisY=titleAxisY[:-2] + ']'
    else:
        features=featuresSensors
        featuresX=fAcc
        featuresY=fGyro
        titleAxisX='Accelerator ['
        titleAxisY='Gyroscope ['
        for item in featuresX:
            titleAxisX=titleAxisX+item+', '
        titleAxisX=titleAxisX[:-2] + ']'
        for item in featuresY:
            titleAxisY=titleAxisY+item+', '
        titleAxisY=titleAxisY[:-2] + ']'
else:
    print('Wrong Answer')
    sys.exit()
#-----------------------------------------------------
```

Παράρτημα Κώδικα 35: Επιλογή Χαρακτηριστικών Αισθητήρων

3. Διαδικασία τυχόν φιλτραρίσματος των δεδομένων για μεμονωμένη επεξεργασία.

```
#---  Filtering  ----------------------------------------------------------------------------
dfFiltered, row_count = filterData(df)
print(f"The Filtered Data have {row_count} Rows!")
if row_count <= 0:
    sys.exit()
df_scaled = dfFiltered[features]
# Find the more appropriate number of samples for silhouette score
score_sample = round(row_count * 5 / 100)
#--------------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 36: Φιλτράρισμα Δεδομένων

Επίσης γίνεται έλεγχος αν υπάρχουν εγγραφές και αν όχι η εφαρμογή τερματίζει.

Το φιλτράρισμα των δεδομένων πραγματοποιείται μέσω της συνάρτησης filterData της οποίας ο κώδικας είναι:

```
def filterData(df):
    flt_s = input("\nFilter application (0/1/2/3)\n"+
                  "0: None\n"+
                  "1: Driver\n"+
                  "2: DayPart\n"+
                  "3: Driver & DayPart\n"+
                  "Enter choice: ")
    print("you chose: " + flt_s + "\n")
    flt = int(flt_s)

    if flt==0:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle.append('All')
    elif flt==1:
        print("\nDataset will be filtered for some driver!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        dfFiltered = df.loc[df['Driver'] == flt1]
        plotTitle.append('Driver'+flt1_s)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['DayPart']))
            print('\nDriver'+flt1_s+' made ' + str(n) + ' roots\n')
    elif flt==2:
        print("\nDataset will be filtered for some part of the Day!\n")
        flt1_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt1_s + "\n")
        flt1 = float(flt1_s)
        dfFiltered = df.loc[df['DayPart'] == flt1]
        if flt1==0.25:
            title1="Morning"
        elif flt1==0.5:
            title1="Afternoon"
        elif flt1==0.75:
            title1="Evening"
        elif flt1==1.0:
            title1="Night"
        else:
            title1="All"
        plotTitle.append('Part of Day '+title1)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['Driver']))
            print('\nThe Part of Day '+flt1_s+' made by ' + str(n) + ' Drivers\n')
    elif flt==3:
        print("\nDataset will be filtered for a compination of Driver AND part of the Day!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        title1 = 'Driver'+flt1_s
        flt2_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt2_s + "\n")
        flt2 = float(flt2_s)
        dfFiltered = df[(df.Driver == flt1) & (df.DayPart == flt2)]
        if flt2==0.25:
            title2="Morning"
        elif flt2==0.5:
            title2="Afternoon"
        elif flt2==0.75:
            title2="Evening"
        elif flt2==1.0:
            title2="Night"
        else:
            title2="All"
        plotTitle.append(title1 + ' and ' + title2)
    else:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle.append('All')

    row_count = len(dfFiltered)

    return dfFiltered, row_count
```

Παράρτημα Κώδικα 37: Συνάρτηση φιλτραρίσματος δεδομένων

Μέσω αυτής της συνάρτησης ο χρήστης έχει την δυνατότητα να επιλέξει να

έχει στην διάθεσή του όλα τα δεδομένα, τα δεδομένα ενός συγκεκριμένου οδηγού, ενός συγκεκριμένου μέρους της ημέρας ή συνδυασμό οδηγού – μέρους της ημέρας.

4. Επιλογή εφαρμογής μεθόδου μετασχηματισμού δεδομένων
5. Εφαρμογή (αν έχει επιλεγεί) μεθόδου ανίχνευσης ακραίων σημείων στα δεδομένα μας και εξομάλυνση αυτών.

```
#---  Outliers Detecting Methods  ------------------------------------------------------
answer_str = input("\nPerform Outliers Detecting technique? (0/1)\n"+
                   "0: None\n"+
                   "1: Winsorize\n"+
                   "Enter choice: ")
print("you chose: " + answer_str + "\n")
answer = int(answer_str)
df_scaled = outliersDetection(features, answer, df_scaled)
#-------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 38: Επιλογή μεθόδου Ανίχνευσης ακραίων Σημείων

Η διαδικασία πραγματοποιείται μέσω της συνάρτησης ouliersDetection:

```
def outliersDetection(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1:
        # Apply Winsorizer to data
        df1 = df
        for column in features:
            df1.loc[:, column] = winsorize(df1[column], limits=[0.01, 0.01])
        plotTitle.append('Winsorizer')
    return df1
```

Παράρτημα Κώδικα 39: Συνάρτηση Ανίχνευσης Ακραίων Σημείων

6. Εφαρμογή (αν έχει επιλεγεί) μεθόδου κανονικοποίησης των δεδομένων μας σε συγκεκριμένο εύρος. Υπάρχει επιλογή τριών διαφορετικών αλγόριθμων κανονικοποίησης (L1, L2, Max) από τους οποίους ο καθένας επιτελεί διαφορετικό τρόπο κανονικοποίησης.

```
#---  Normalization Methods  -----------------------------------------------------
answer_str = input("\nPreferred Normalization technique (0/1/2/3)\n"+
                   "0: None\n"+
                   "1: Normalize (L1)\n"+
                   "2: Normalize (L2)\n"+
                   "3: Normalize (Max)\n"+
                   "Enter choice: ")
print("you chose: " + answer_str + "\n")
answer = int(answer_str)
df_scaled = normalization(features, answer, df_scaled)
#-------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 40: Επιλογή μεθόδου Κανονικοποίησης

Η διαδικασία εκτελείται μέσω της συνάρτησης normalization:

```
def normalization(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1 or answer==2 or answer==3:
        if answer==1: norm='l1'
        if answer==2: norm='l2'
        if answer==3: norm='max'
        # Apply Normalization to data
        df1 = normalize(df, norm=norm)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Normalization ' + norm)
    return df1
```

Παράρτημα Κώδικα 41: Συνάρτηση Κανονικοποίησης Δεδομένων

7. Εφαρμογή (αν έχει επιλεγεί) μεθόδου κλιμάκωσης χαρακτηριστικών των δεδομένων μας έτσι ώστε ανόμοιες μετρήσεις από διαφορετικούς αισθητήρες να κλιμακωθούν σε ενιαία κλίμακα. Ο χρήστης έχει την επιλογή μεταξύ τριών διαφορετικών μεθόδων:

```
#---  Features Scaling Methods  ------------------------------------------------------
answer_str = input("\nPreferred Features Scaling technique (0/1/2/3)\n"+
                   "0: None\n"+
                   "1: StandardScaler\n"+
                   "2: MinMaxScaler\n"+
                   "3: RobustScaler\n"+
                   "Enter choice: ")
print("you chose: " + answer_str + "\n")
answer = int(answer_str)

df_scaled = featuresScaling(features, answer, df_scaled)
#-------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 42: Επιλογή Μεθόδου Κλιμάκωσης Χαρακτηριστικών

Η συνάρτηση που εκτελεί την διαδικασία είναι η featuresScaling:

```
def featuresScaling(features, preProp, df):
    if preProp==0:
        df1 = df
    elif preProp==1:
        # Standardize the features for clustering
        scaler = StandardScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Standardize')
    elif preProp==2:
        # Apply MinMaxScaler to data
        scaler = MinMaxScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('MinMaxScaler')
    elif preProp==3:
        # Apply RobustScaler to data
        scaler = RobustScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('RobustScaler')
    return df1
```

Παράρτημα Κώδικα 43: Συνάρτηση Μεθόδου Κλιμάκωσης Χαρακτηριστικών

8. Επανάληψη βημάτων 4-7 έως ότου αποφασίσει ο χρήστης
9. Εφαρμογή μεθόδου μείωσης διαστάσεων σε δύο έτσι ώστε να είναι πιο αποδοτική η οπτική αξιολόγηση, καθώς και η πολυπλοκότητα επεξεργασίας. Αυτό επιτυγχάνεται μέσω της συνάρτησης methodPCA:

```
#---  PCA Method  ----------------------------------------
dfFinal = methodPCA(df_scaled, featuresX, featuresY, 1)
#---------------------------------------------------------
```

Παράρτημα Κώδικα 44: Μείωση Διαστάσεων Δεδομένων

```python
def methodPCA(df_scaled, featuresX, featuresY, num):
    # Create PCA object for Axis X
    if len(featuresX)>1:
        pca_X = PCA(n_components=num)
        pcaX = pca_X.fit_transform(df_scaled[featuresX])
    else:
        pcaX = df_scaled[featuresX].values

    # Create PCA object for Axis Y
    if len(featuresY)>1:
        pca_Y = PCA(n_components=num)
        pcaY = pca_Y.fit_transform(df_scaled[featuresY])
    else:
        pcaY = df_scaled[featuresY].values

    # Concatenate PCA results
    dfFinal = pd.DataFrame({
        'axisX': pcaX.flatten(),
        'axisY': pcaY.flatten()
    })
    plotTitle.append('PCA')

    return dfFinal
```

Παράρτημα Κώδικα 45: Συνάρτηση Μείωσης Διαστάσεων Δεδομένων

Η μείωση των έξι διαστάσεων σε δύο πραγματοποιείται μειώνοντας πρώτα τις τρεις διαστάσεις που αποτελούν τα χαρακτηριστικά του αισθητήρα Accelerator (AccX, AccY, AccZ) σε μία διάσταση (Acc) και κατόπιν τις τρεις διαστάσεις που αποτελούν τα χαρακτηριστικά του αισθητήρα Gyroscope (GyroX, GyroY, GyroZ) σε μία διάσταση (Gyro). Με αυτό τον τρόπο έχουμε συμπτύξει τα δεδομένα του Accelerator τα οποία αφορούν αυτόν και μόνο τον αισθητήρα σε μια διάσταση, και τα δεδομένα του Gyroscope σε μια άλλη. Αυτό θα μας βοηθήσει να έχουμε καλύτερη οπτική απεικόνιση των δεδομένων μας χρησιμοποιώντας ένα διάγραμμα όπως θα δούμε παρακάτω, καθώς και επιτυγχάνουμε ταχύτερη επεξεργασία των αλγορίθμων.

10. Επιλογή εφαρμογής αξιολόγησης ποιότητας συσταδοποίησης. Αν επιλεγεί να γίνει εκτελούνται τα βήματα 11-16 και έπειτα εκτελείται το βήμα 19. Σε αντίθετη περίπτωση τα βήματα 11-16 παραλείπονται και εκτελείται το βήμα

17.

```
#---  Determine the optimal number of clusters using evaluation  -------------------------
eval = input("Use evaluation to determine the optimal number of clusters? (y/n) :")
print("You entered: " + eval + "\n")
if eval=='y' or eval=='Y':
    k_optimal, cluster_labels, cluster_centers = evaluation()
else:
    k_optimal = int(input("Enter the optimal Clusters number: "))
    #---  Apply K-means clustering with the optimal number of clusters  -------------------
    kmeans = KMeans(n_clusters=k_optimal, init="k-means++", random_state=45)
    cluster_labels = kmeans.fit_predict(dfFinal)
    cluster_centers = kmeans.cluster_centers_
    #-------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 46: Επιλογή Εκτέλεσης Αξιολόγησης Ποιότητας Συσταδοποίησης

11. Επιλογή αριθμού συστάδων για εφαρμογή των δεδομένων του σεναρίου στον αλγόριθμο K-means και υπολογισμού μετρικών αξιολόγησης ποιότητας συσταδοποίησης.
12. Εφαρμογή του αλγορίθμου μηχανικής μάθησης χωρίς επίβλεψη για συσταδοποίηση (clustering) K-means για τον συγκεκριμένο αριθμό συστάδων.
13. Εκτέλεση υπολογισμού και καταχώρησης των τεσσάρων μετρικών αξιολόγησης ποιότητας συσταδοποίησης για τα αποτελέσματα του αλγόριθμου K-means
14. Εκτέλεση βημάτων 11-13 για το εύρος αριθμού συστάδων 2-11
15. Οπτικοποίηση των τεσσάρων μετρικών για το εύρος αριθμού συστάδων 2-11
16. Επιλογή του βέλτιστου αριθμού συστάδων από τον χρήστη βάση του βήματος 15.
17. Καταχώρηση από τον χρήστη του βέλτιστου αριθμού (k) συστάδων έτσι ώστε να διοχετευτεί στον κώδικα εκτέλεσης του αλγόριθμου K-means

```
#---  Based on the elbow method, choose the optimal number of clusters  ------------------
k_str = input("Enter the number of optimal Clusters:")
print("Number of Clusters is: " + k_str + "\n")
k_optimal = int(k_str)
#-----------------------------------------------------------------------------------------
```

Παράρτημα Κώδικα 47: Καταχώρηση βέλτιστου αριθμού (k) Συστάδων

18. Εφαρμογή του αλγορίθμου μηχανικής μάθησης χωρίς επίβλεψη για συσταδοποίηση (clustering) K-means.
19. Αποτύπωση διαγράμματος δύο διαστάσεων με τις συστάδες που παρήχθησαν και τα κέντρα τους, για οπτική αξιολόγηση. Στην οπτικοποίηση της συσταδοποίησης που διενεργήθηκε εμφανίζεται με διαφορετικό χρώμα η κάθε συστάδα, καθώς και τα κέντρα των συστάδων με κόκκινο χρώμα και σύμβολο

Χ. Προτού όμως γίνει η οπτικοποίηση γίνεται καταχώρηση ως χαρακτηριστικό (Cluster) στο dataframe των δεδομένων μας ο δείκτης συστάδας του κάθε σημείου. Οι συντεταγμένες των κέντρων συστάδων αποθηκεύονται ως εικόνα με συγκεκριμένη χαρακτηριστική ονοματολογία. Επίσης υπάρχει η δυνατότητα από τον χρήστη να ονοματίσει τις συστάδες για το διάγραμμα.

```
#---  Add the cluster labels to the Final DataFrame
dfFinal['Cluster'] = cluster_labels

#---  Get the cluster centers
cluster_centers_combined = pd.DataFrame({
    'Acc': kmeans.cluster_centers_[:, 0],
    'Gyro': kmeans.cluster_centers_[:, 1]
})

print("\nClusters Centers Values:\n", cluster_centers_combined)
print("\n")
pltTitle = createPlotTitle(plotTitle)
pltTitle = 'ClustersCenters (' + pltTitle + ')'
dfi.export(cluster_centers_combined, 'C:/datasets/plots/' + pltTitle + '.png')

#--- Name the Clusters for the Plot  ------------------------------------------------------
do_names = input("\nDo you want to name the Clusters? (y/n):")
if do_names=='y':
    sorted_pairs = sorted(enumerate(zip(cluster_centers_combined['Acc'],
                                        cluster_centers_combined['Gyro'])),
                                        key=lambda pair: abs(pair[1][0]) + abs(pair[1][1]))
    labels = ['Normal', 'Aggresive', 'Dangerous']
    clabels=[sorted_pairs[0][0], sorted_pairs[1][0], sorted_pairs[2][0]]
    print(clabels)

    dfFinal['Cluster'] = np.where(dfFinal['Cluster']==clabels[0],labels[0],
                                  np.where(dfFinal['Cluster']==clabels[1],labels[1],
                                           np.where(dfFinal['Cluster']==clabels[2],labels[2],'')))
#-------------------------------------------------------------------------------------------
```

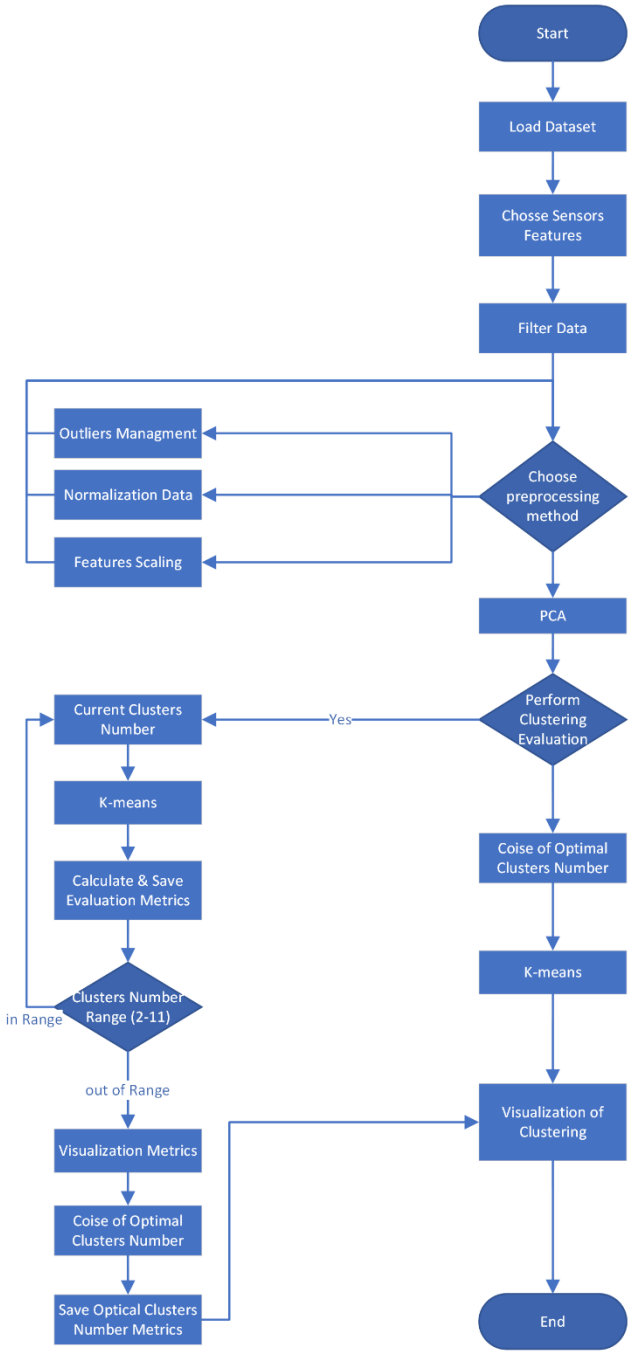
Παράρτημα Κώδικα 48: Καταχώρηση χαρακτηριστικού Cluster στο Dataset

```
#---  Plot the data points after PCA and K-means clustering  ----------------------------
sns.set_style="whitegrid"
sns.scatterplot(data=dfFinal, x='axisX', y='axisY', hue='Cluster', palette=palette)
plt.scatter(cluster_centers_combined['axisX'], cluster_centers_combined['axisY'],
            c='red', marker='X', s=100, label='Centroids')
# Annotating the cluster centers with their coordinates
for i, (x, y) in enumerate(zip(cluster_centers_combined['axisX'], cluster_centers_combined['axisY'])):
    if i % 2 == 0:
        plt.text(x, y + 0.1, f'({x:.3f}, {y:.3f})',
                 fontsize=10, color='black', ha='left', va='bottom', weight='bold')
    else:
        plt.text(x, y + 0.1, f'({x:.3f}, {y:.3f})',
                 fontsize=10, color='black', ha='right', va='top', weight='bold')

pltTitle = createPlotTitle(plotTitle)
plt.title(pltTitle)
plt.xlabel(titleAxisX)
plt.ylabel(titleAxisY)

# Get legend handles and labels
handles, labels = plt.gca().get_legend_handles_labels()
# Format cluster labels in legend
formatted_labels = [f"{label} Cluster" if label != 'Centroids' else label for label in labels]
# Update legend with formatted labels
plt.legend(handles, formatted_labels, loc='upper right', fontsize='6')

plt.savefig("c:/datasets/plots/" + pltTitle + ".png")
plt.show()
```

Παράρτημα Κώδικα 49: Δημιουργία Διαγράμματος 2D χρωματισμένων Συστάδων με τα κέντρα τους

Το διάγραμμα αποθηκεύεται ως εικόνα με συγκεκριμένη χαρακτηριστική ονοματολογία.

Στο παρακάτω διάγραμμα ροής, γίνεται παρουσίαση των παραπάνω βημάτων εκτέλεσης:



DriverPredict_(Kmeans).py

Σχήμα 38: Διάγραμμα Ροής DriverPredict_(Kmeans).py

Ολόκληρος ο κώδικας της υλοποίησης του αλγόριθμου K-means για την εφαρμογή μας βρίσκεται στην τοποθεσία:

https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link με όνομα **DriverPredict_(K-means).py**

Σενάρια Εκτέλεσης

Η οπτικοποίηση των δεδομένων λαμβάνει μέρος σε διάγραμμα δύο διαστάσεων όπου ο άξονας Accelerator υποδηλώνει τις τιμές του αντίστοιχου αισθητήρα και ο άξονας Gyroscope υποδηλώνει τις αντίστοιχες τιμές.
Όσο πιο μεγάλες θετικές τιμές έχει ο άξονας Accelerator, υποδηλώνει μεγαλύτερη επιτάχυνση και όσο πιο μεγάλες αρνητικές τιμές έχει αντίστοιχα μεγαλύτερη επιβράδυνση. Οι τιμές κοντά στο μηδέν υποδηλώνουν ελάχιστη χωρίς διακύμανση ταχύτητα.
Όσο πιο μεγάλες θετικές ή αρνητικές τιμές έχει ο άξονας Gyroscope, υποδηλώνει μεγαλύτερη γωνία στροφής προς την μία ή την άλλη κατεύθυνση. Οι τιμές κοντά στο μηδέν υποδηλώνουν σταθερή πορεία χωρίς αισθητή διακύμανση στροφών.

Επομένως ένα σημείο για παράδειγμα που έχει accelerator = 0.10 και gyroscope = 0.5 υποδεικνύει μεγάλη επιτάχυνση σε απότομη στροφή.
Ένα σημείο με accelerator = 0 και gyroscope = 0.3 υποδεικνύει καθόλου επιτάχυνση σε μέτρια στροφή.
Ένα σημείο με accelerator = -0.10 και gyroscope = -0.3 υποδεικνύει απότομη επιβράδυνση σε μέτρια στροφή.
Ένα σημείο με accelerator = 0 και gyroscope = 0 υποδεικνύει ακινητοποιημένο όχημα.

Στην υλοποίηση της εφαρμογής έχουμε ενσωματώσει τις διαδικασίες προ-επεξεργασίας και μετασχηματισμού δεδομένων που αναφέραμε στη ανάλυση και σχεδιασμό της εφαρμογής (Winsorize, L1, L2, Max, StandardScaler, MinMaxScaler, RobustScaler), ο καθένας για τους λόγους που προαναφέρθηκαν.

Εκτέλεση 1ης Ομάδας Σεναρίων

Θα εκτελεστεί ένας αριθμός σεναρίων (32 σενάρια), ο οποίος θα καλύπτει την εφαρμογή της κάθε διαδικασίας μεμονωμένα αλλά και συνδυαστικά με τις άλλες, καλύπτοντας όλους τους συνδυασμούς εφαρμογής διαδικασιών ανίχνευσης ακραίων σημείων, κανονικοποίησης δεδομένων και κλιμάκωσης χαρακτηριστικών.

Η σειρά συνδυασμού θα είναι:

Διαχ. Ακραίων Σημείων → Κανονικοποίηση → Κλιμάκωση Χαρακτηριστικών

Σε κάθε σενάριο θα εκτελεστεί η διαδικασία μείωσης διαστάσεων (PCA), η οποία μειώνει τις 3 διαστάσεις των δεδομένων του επιταχυνσιόμετρου σε μία και τις 3 διαστάσεις του γυροσκοπίου σε μία.

Επίσης τα σενάρια θα εκτελεστούν στο σύνολο των δεδομένων των αισθητήρων (178360 εγγραφές).

Τα σενάρια υλοποίησης που θα εκτελεστούν είναι τα εξής:

No	Senario	DATA	ODR	ND	FS	PCA	ULA
0	0000000	All	-	-	-	√	K-means
1	1000000	All	Winsorize	-	-	√	K-means
2	0100000	All	-	L1	-	√	K-means
3	0010000	All	-	L2	-	√	K-means
4	0001000	All	-	Max	-	√	K-means
5	0000100	All	-	-	StandarScaler	√	K-means
6	0000010	All	-	-	MinMaxScaler	√	K-means
7	0000001	All	-	-	RobustScaler	√	K-means
8	1100000	All	Winsorize	L1	-	√	K-means
9	1010000	All	Winsorize	L2	-	√	K-means
10	1001000	All	Winsorize	Max	-	√	K-means
11	1000100	All	Winsorize	-	StandarScaler	√	K-means
12	1000010	All	Winsorize	-	MinMaxScaler	√	K-means
13	1000001	All	Winsorize	-	RobustScaler	√	K-means
14	1100100	All	Winsorize	L1	StandarScaler	√	K-means
15	1100010	All	Winsorize	L1	MinMaxScaler	√	K-means
16	1100001	All	Winsorize	L1	RobustScaler	√	K-means

17	1010100	All	Winsorize	L2	StandarScaler	√	K-means
18	1010010	All	Winsorize	L2	MinMaxScaler	√	K-means
19	1010001	All	Winsorize	L2	RobustScaler	√	K-means
20	1001100	All	Winsorize	Max	StandarScaler	√	K-means
21	1001010	All	Winsorize	Max	MinMaxScaler	√	K-means
22	1001001	All	Winsorize	Max	RobustScaler	√	K-means
23	0100100	All	-	L1	StandarScaler	√	K-means
24	0100010	All	-	L1	MinMaxScaler	√	K-means
25	0100001	All	-	L1	RobustScaler	√	K-means
26	0010100	All	-	L2	StandarScaler	√	K-means
27	0010010	All	-	L2	MinMaxScaler	√	K-means
28	0010001	All	-	L2	RobustScaler	√	K-means
29	0001100	All	-	Max	StandarScaler	√	K-means
30	0001010	All	-	Max	MinMaxScaler	√	K-means
31	0001001	All	-	Max	RobustScaler	√	K-means

Η κωδικοποίηση του scenario υποδηλώνει ποιες διαδικασίες θα εκτελεστούν και με ποια σειρά.

Η σειρά είναι:

Winsorize →L1 →L2 →Max →StandardScaler →MinMaxScaler →RobustScaler

(π.χ. το scenario 22 1001001 έχει άσσο στην $1^{η}$ $4^{η}$ και $7^{η}$ θέση άρα θα εκτελέσει Winsorize→Max→RobustScaler).

Η κωδικοποίηση των σεναρίων βρίσκεται σε αρχείο με όνομα ***executionSenarios.csv*** και βρίσκεται στην τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

Το αρχείο αυτό φορτώνεται και χρησιμοποιείται μέσα από τον κώδικα εκτέλεσης των σεναρίων.

Για το κάθε σενάριο θα εκτελεστούν οι μετρικές αξιολόγησης που αναφέραμε (Inertia, Silhouette Score, Calinski-Harabasz Index, Davies-Bouldin Index). Βάση ενός διαγράμματος των τεσσάρων μετρικών ο χρήστης θα μπορεί μέσω οπτικοποίησης να αξιολογήσει ποιος είναι ο ιδανικός αριθμών συστάδων που όταν εφαρμόζεται στον αλγόριθμο K-means παρέχει την καλύτερη ποιότητα συσταδοποίησης.

Η επιλογή του χρήστη για το κάθε σενάριο αποθηκεύει σε αρχείο με όνομα ***clustersMetrics.csv*** (τοποθεσία στο https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link) το σενάριο, την περιγραφή του σεναρίου, τον βέλτιστο αριθμό συστάδων, τις τιμές των τεσσάρων μετρικών. Μετά την εκτέλεση όλων των σεναρίων υπολογίζεται και καταχωρείται ένα τελικό σκορ για το κάθε σενάριο, που θα κρίνει ποιος συνδυασμός είναι πιο αποδοτικός.

Το τελικό σκορ υπολογίζεται αφού εκτελεστεί κανονικοποίηση των τεσσάρων μετρικών στο εύρος τιμών (0, 1), έτσι ώστε η κάθε μετρική να έχει το ίδιο βάρος στον υπολογισμό του τελικού σκορ. Το μηδέν αντιπροσωπεύει την μικρότερη τιμή και ο άσσος την μεγαλύτερη.

Η κανονικοποίηση για την κάθε τιμή της κάθε μετρικής γίνεται ως εξής:

$$(Τιμή\ Μετρικής - Ελάχιστη\ Τιμή\ Μετρικής)/(Μέγιστη\ Τιμή\ Μετρικής - Ελάχιστη\ Τιμή\ Μετρικής)$$

Αφού γίνει η κανονικοποίηση όλων των μετρικών, υπολογίζεται το τελικό σκορ για το κάθε σενάριο λαμβάνοντας υπόψη ότι για τις μετρικές inertia και Davies-Bouldin Index η ελάχιστη τιμή είναι η καλύτερη, ενώ για τις άλλες δύο ότι η μέγιστη τιμή είναι η καλύτερη, ως εξής:

$$Silhouette + Calinski - Inertia - Davis$$

Όσο μεγαλύτερο το τελικό σκορ για το κάθε σενάριο, τόσο καλύτερης ποιότητας είναι η συσταδοποίηση που επιτυγχάνει. Το μεγαλύτερο σκορ λοιπόν που μπορεί να επιτύχει κάποιο σενάριο είναι το +2, ενώ το χειρότερο είναι -2.

Ο κώδικας εκτέλεσης των σεναρίων έχει όνομα ***DriverPredictSenarios_(K-means).py*** και βρίσκεται στην τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

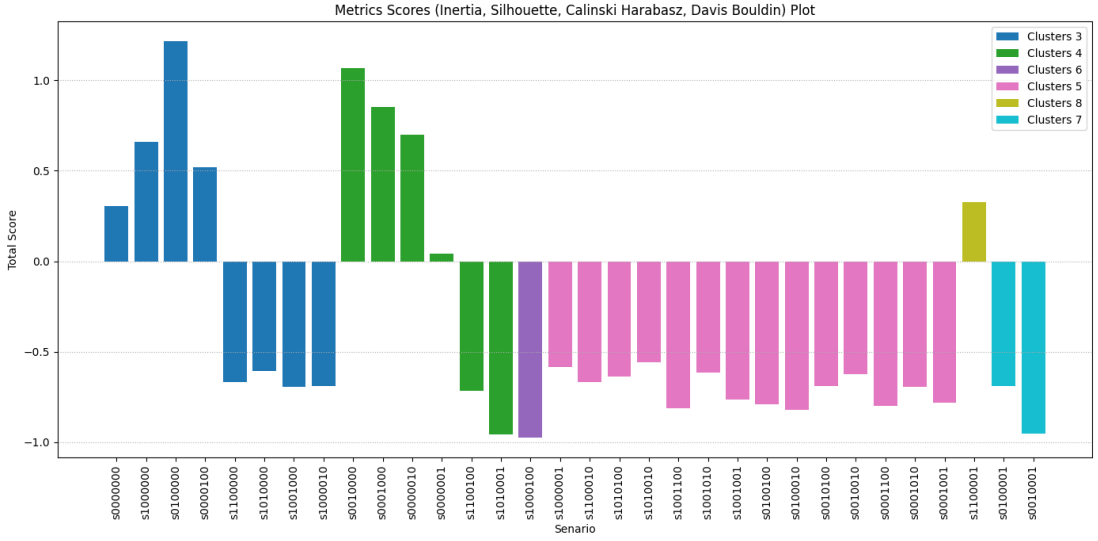
Αφού εκτελέστηκαν όλα τα σενάρια, τα αποτελέσματα που λάβαμε είναι τα εξής:

Senario	Title	Clusters	Inertia	Silhouette	Calinski	Davis	TotalScore
s0000000	All-PCA-K-means	3	274274,703	0,551	500836,874	0,571	0,30546361
s1000000	All-Winsorizer-PCA-K-means	3	207788,373	0,561	642145,892	0,552	0,659760905
s0100000	All-Normalization l1-PCA-K-means	3	842,444	0,604	813584,832	0,555	1,213538058
s0010000	All-Normalization l2-PCA-K-means	4	1396,397	0,568	669119,985	0,512	1,066960772
s0001000	All-Normalization	4	2224,419	0,558	538978,801	0,515	0,850967607

	max-PCA-K-means						
s0000100	All-Standardize-PCA-K-means	3	180219,613	0,683	171452,489	0,515	0,5206942
s0000010	All-MinMaxScaler-PCA-K-means	4	2552,817	0,551	470260,793	0,526	0,699173028
s0000001	All-RobustScaler-PCA-K-means	4	980696,457	0,785	243161,606	0,528	0,041173886
s1100000	All-Winsorizer-Normalization l1-PCA-K-means	3	9941,914	0,48	202473,176	0,753	-0,667145108
s1010000	All-Winsorizer-Normalization l2-PCA-K-means	3	26599,916	0,49	221521,27	0,746	-0,605723412
s1001000	All-Winsorizer-Normalization max-PCA-K-means	3	46192,649	0,486	215403,518	0,76	-0,694348477
s1000100	All-Winsorizer-Standardize-PCA-K-means	6	75667,943	0,431	183956,974	0,775	-0,97379813
s1000010	All-Winsorizer-MinMaxScaler-PCA-K-means	3	46219,53	0,487	215462,235	0,759	-0,687986335
s1000001	All-Winsorizer-RobustScaler-PCA-K-means	5	68307,985	0,471	190148,868	0,701	-0,58619008
s1100100	All-Winsorizer-Normalization l1-Standardize-PCA-K-means	4	114654,895	0,464	176689,068	0,714	-0,71697765
s1100010	All-Winsorizer-Normalization l1-MinMaxScaler-PCA-K-means	5	6958,688	0,45	213215,853	0,734	-0,666888937
s1100001	All-Winsorizer-Normalization l1-RobustScaler-PCA-K-means	8	62988,697	0,428	908334,565	0,686	0,326049692
s1010100	All-Winsorizer-Normalization l2-Standardize-PCA-K-means	5	90977,578	0,473	224975,068	0,724	-0,637163713
s1010010	All-Winsorizer-Normalization l2-MinMaxScaler-PCA-K-means	5	16536,69	0,471	234906,114	0,725	-0,556826202
s1010001	All-Winsorizer-Normalization l2-RobustScaler-PCA-K-means	4	74195,539	0,454	206667,164	0,797	-0,954243365
s1001100	All-Winsorizer-Normalization max-Standardize-PCA-K-means	5	101148,687	0,453	205340,839	0,747	-0,810913114
s1001010	All-Winsorizer-Normalization max-MinMaxScaler-PCA-K-means	5	30430,893	0,462	223674,352	0,726	-0,614967207
s1001001	All-Winsorizer-Normalization max-RobustScaler-PCA-K-means	5	60096,652	0,447	210093,446	0,742	-0,761830287
s0100100	All-Normalization l1-Standardize-PCA-K-means	5	92270,084	0,453	208702,989	0,745	-0,790271751
s0100010	All-Normalization l1-MinMaxScaler-PCA-K-means	5	10186,809	0,438	214110,095	0,768	-0,821881571

s0100001	All-Normalization l1-RobustScaler-PCA-K-means	7	53299,196	0,467	189659,318	0,731	-0,6880047
s0010100	All-Normalization l2-Standardize-PCA-K-means	5	90348,596	0,466	225424,944	0,734	-0,690606849
s0010010	All-Normalization l2-MinMaxScaler-PCA-K-means	5	16536,135	0,464	232699,018	0,738	-0,625042696
s0010001	All-Normalization l2-RobustScaler-PCA-K-means	7	41971,14	0,443	202586,694	0,795	-0,950688687
s0001100	All-Normalization max-Standardize-PCA-K-means	5	99659,486	0,45	207727,226	0,742	-0,797014304
s0001010	All-Normalization max-MinMaxScaler-PCA-K-means	5	30382,495	0,45	220805,724	0,738	-0,694529449
s0001001	All-Normalization max-RobustScaler-PCA-K-means	5	61088,499	0,445	206366,97	0,744	-0,780519397

Οπτικοποιήσαμε τα αποτελέσματα μέσω γραφήματος μπάρας, αναπαριστώντας το κάθε σενάριο σε σχέση με το τελικό σκορ που επιτεύχθηκε. Με χρώμα και ετικέτες αναπαρίστανται οι βέλτιστοι αριθμοί συστάδων για το κάθε σενάριο. Στο γράφημα αποτυπώνεται η ποιότητα συσταδοποίησης του κάθε σεναρίου.

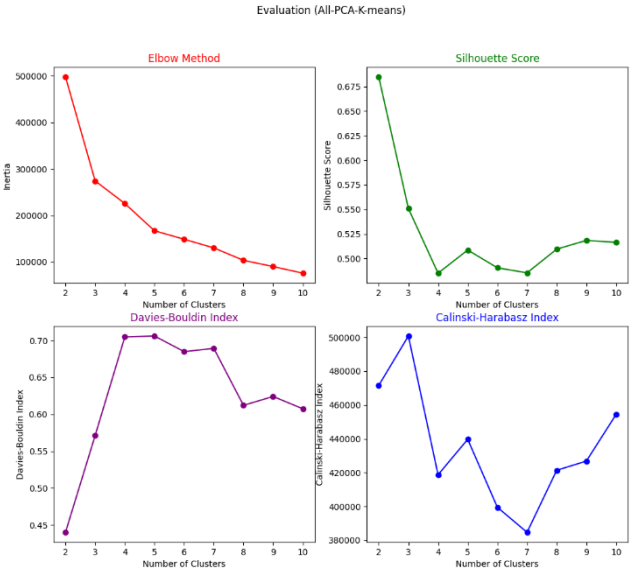


Σχήμα 39: Γράφημα ποιότητας συσταδοποίησης 1ης ομάδας σεναρίων

Στο πρώτο σενάριο (s0000000) δεν γίνεται εφαρμογή καμίας μεθόδου ανίχνευσης ακραίων σημείων, κανονικοποίησης ή κλιμάκωσης χαρακτηριστικών και βάση των μετρικών αξιολόγησης ποιότητας συσταδοποίησης είχαμε το εξής αποτέλεσμα:

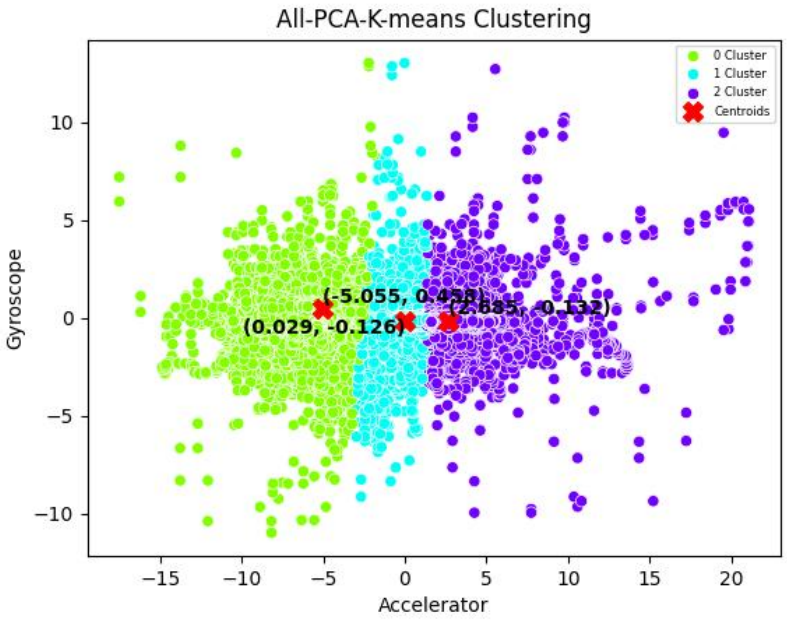
Σενάριο-0 (s0000000)

Μετρήσεις αξιολόγησης ποιότητας συσταδοποίησης:



Inertia:	Silhouette score:	Davis-Bouldin Index:	Calinski-Harabasz Index:
274274.703	0.551	0.571	500836.874

Αρ. Συστάδων:	Τελικό Σκορ:	Clusters Centers Values:
3	0.30546361	Cluster 1 --> Acc: 0.029 -- Gyro: -0.126 Cluster 2 --> Acc: 2.685 -- Gyro: -0.132 Cluster 0 --> Acc: -5.055 -- Gyro: 0.458



Παρατηρήσεις – Συμπεράσματα Σεναρίου 0:

Βάση των μετρικών η καλύτερη επιλογή για να επιτευχθεί η καλύτερη δυνατόν απόδοση ήταν ο αριθμός 3 συστάδων.

Από τις επιμέρους μετρικές, καθώς και από το τελικό σκορ συμπεραίνεται μια μέτρια ποιότητας συσταδοποίηση. Πιο αραιά τα σημεία γύρω από τα κεντροειδή τους, όχι τόσο καλά διαχωρισμένες συστάδες με σχετική επικάλυψη και μεγάλη διακύμανση εντός συστάδων.

Από την οπτικοποίηση της συσταδοποίησης συμπεραίνεται η ύπαρξη ακραίων σημείων που δεν βοηθούν στην σωστή συσταδοποίηση, διακύμανση εντός συστάδων και σχετικά κοντινή απόσταση των κεντροειδών μεταξύ τους.

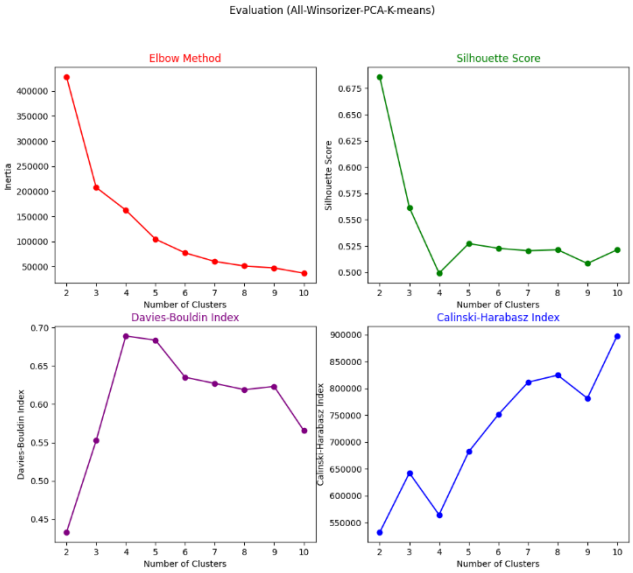
Ο διαχωρισμός των συστάδων δεν είναι ικανοποιητικός για την λήψη σωστών συμπερασμάτων, καθώς δεν μπορεί να αποτυπωθεί με ασφάλεια η διαφοροποίηση στην οδηγική συμπεριφορά. Φαίνεται σαν ο διαχωρισμός να έχει γίνει με βάση μόνο των δεδομένων του επιταχυνσιόμετρου, διαχωρίζοντας σε 3 συστάδες την διαφορετική ταχύτητα. Για παράδειγμα θα μπορούσε να συμπεράνει κάποιος ότι η συστάδα 1 είναι φυσιολογική οδήγηση (μικρότερες τιμές κεντροειδούς), η συστάδα 2 είναι επιθετική ή επικίνδυνη οδήγηση με αυξημένη ταχύτητα και η συστάδα 0 είναι επιθετική ή επικίνδυνη οδήγηση με μεγάλη επιβράδυνση.

Βάση των παραπάνω, χρίζεται αναγκαία η εφαρμογή κάποιας μεθόδου ανίχνευσης ακραίων σημείων, κανονικοποίησης δεδομένων, κλιμάκωσης χαρακτηριστικών ή συνδυασμός αυτών.

Στο επόμενο σενάριο (s1000000) γίνεται εφαρμογή της μεθόδου ανίχνευσης ακραίων σημείων winsorize και βάση των μετρικών αξιολόγησης ποιότητας συσταδοποίησης είχαμε το εξής αποτέλεσμα:

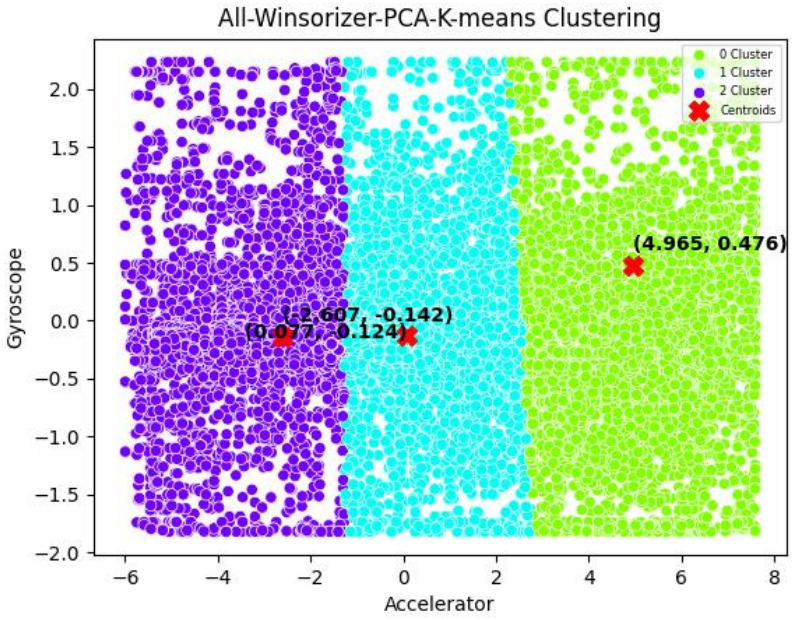
Σενάριο-1 (s1000000)

Μετρήσεις αξιολόγησης ποιότητας συσταδοποίησης:



Inertia:	Silhouette score:	Davis-Bouldin Index:	Calinski-Harabasz Index:
207788.373	0.561	0.552	642145.892

Αρ. Συστάδων:	Τελικό Σκορ:	
3	0.659760905	Clusters Centers Values: Cluster 1 --> Acc: 0.077 -- Gyro: -0.124 Cluster 2 --> Acc: -2.607 -- Gyro: -0.142 Cluster 0 --> Acc: 4.965 -- Gyro: 0.476



Παρατηρήσεις – Συμπεράσματα Σεναρίου 1:

Βάση των μετρικών η καλύτερη επιλογή για να επιτευχθεί η καλύτερη δυνατόν απόδοση ήταν ο αριθμός 3 συστάδων.

Σε σύγκριση με το σενάριο 0, παρατηρούμε μια μικρή βελτίωση όσον αφορά την ποιότητα συσταδοποίησης που πραγματοποιείται. Ισχύουν οι ίδιες παρατηρήσεις για τις μετρικές του σεναρίου 0.

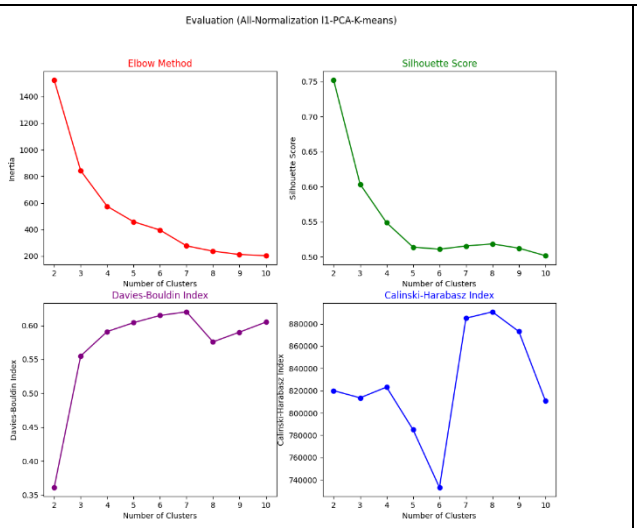
Η διαφορά έγκειται στην οπτικοποίηση της συσταδοποίησης. Δεν υπάρχουν ακραία σημεία, τα οποία έχουν αντικατασταθεί με τα πλησιέστερα μη ακραία σημεία. Τα κέντρα των συστάδων βρίσκονται σε ικανοποιητική απόσταση.

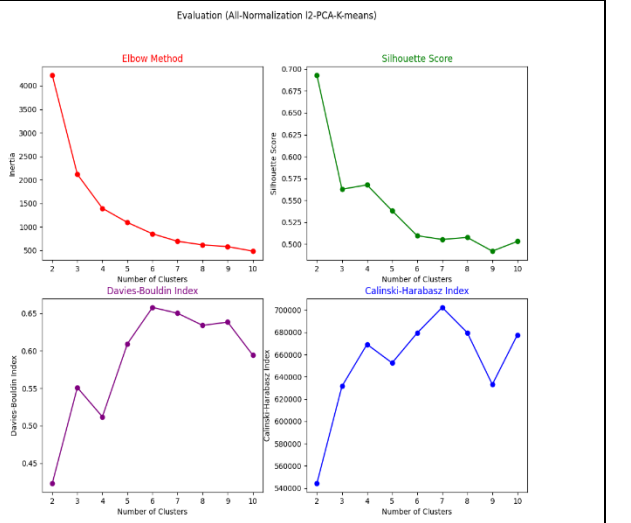
Και σε αυτήν την περίπτωση, ο διαχωρισμός των συστάδων δεν είναι ικανοποιητικός για την λήψη σωστών συμπερασμάτων, καθώς δεν μπορεί να αποτυπωθεί με ασφάλεια η διαφοροποίηση στην οδηγική συμπεριφορά. Φαίνεται σαν ο διαχωρισμός να έχει γίνει με βάση μόνο των δεδομένων του επιταχυνσιόμετρου, διαχωρίζοντας σε 3 συστάδες την διαφορετική ταχύτητα. Για παράδειγμα θα μπορούσε να συμπεράνει κάποιος ότι η συστάδα 1 είναι φυσιολογική οδήγηση (μικρότερες τιμές κεντροειδούς), η συστάδα 2 είναι επιθετική ή επικίνδυνη οδήγηση με αυξημένη ταχύτητα και η συστάδα 0 είναι επιθετική ή επικίνδυνη οδήγηση με μεγάλη επιβράδυνση.

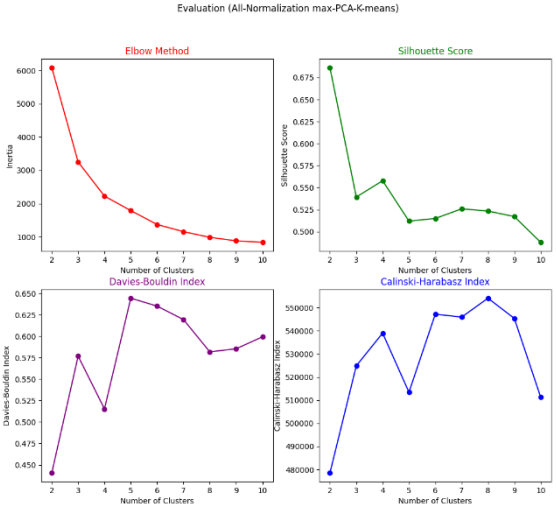
Βάση των παραπάνω, διακρίνεται μια βελτίωση της ποιότητας της συσταδοποίησης χωρίς όμως να είναι αρκετή για ασφαλή συμπεράσματα. Χρίζεται αναγκαία η εφαρμογή κάποιας συνδυαστικής μεθόδου κανονικοποίησης δεδομένων, κλιμάκωσης χαρακτηριστικών ή συνδυασμός αυτών. Από τα επόμενα σενάρια θα μπορέσει να βγει συμπέρασμα αν χρειάζεται η εφαρμογή της μεθόδου Winsorize ή μπορεί να καλυφθεί η διαχείριση ακραίων σημείων από μεθόδους κανονικοποίησης ή κλιμάκωσης χαρακτηριστικών.

Στα επόμενα 3 σενάρια (s0100000, s0010000, s0001000) γίνεται εφαρμογή των μεθόδων κανονικοποίησης δεδομένων (L1, L2, Max) και βάση των μετρικών αξιολόγησης ποιότητας συσταδοποίησης είχαμε τα εξής αποτελέσματα:

Σενάρια-2-3-4 (s0100000, s0010000, s00010000)

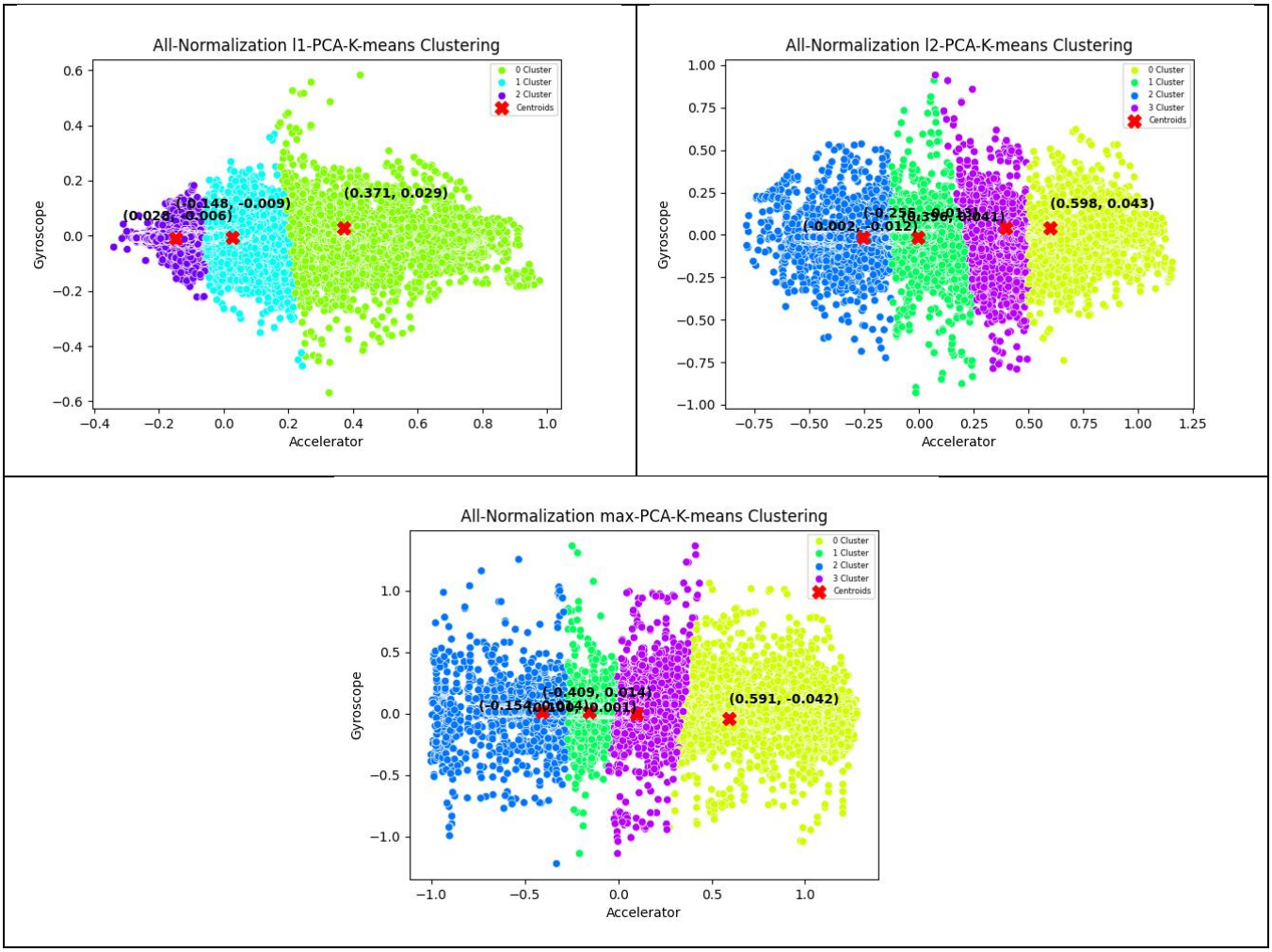






Inertia (L1-L2-Max):	Silhouette score (L1-L2-Max):	Davis-Bouldin Index (L1-L2-Max):	Calinski-Harabasz Index (L1-L2-Max):
842.444	0.604	0.555	813584.832
1396.397	0.568	0.512	669119.985
2224.419	0.558	0.515	538978.801

Αρ. Συστάδων (L1-L2-Max):	Τελικό Σκορ (L1-L2-Max):	
3	1.213538058	Clusters Centers Values: Cluster 1 --> Acc: 0.028 -- Gyro: -0.006 Cluster 2 --> Acc: -0.148 -- Gyro: -0.009 Cluster 0 --> Acc: 0.371 -- Gyro: 0.029
4	1.066960772	Clusters Centers Values: Cluster 1 --> Acc: -0.002 -- Gyro: -0.012 Cluster 2 --> Acc: -0.255 -- Gyro: -0.013 Cluster 3 --> Acc: 0.396 -- Gyro: 0.041 Cluster 0 --> Acc: 0.598 -- Gyro: 0.043
4	0.850967607	Clusters Centers Values: Cluster 3 --> Acc: 0.100 -- Gyro: -0.001 Cluster 1 --> Acc: -0.154 -- Gyro: 0.014 Cluster 2 --> Acc: -0.409 -- Gyro: 0.014 Cluster 0 --> Acc: 0.591 -- Gyro: -0.042



Παρατηρήσεις – Συμπεράσματα Σεναρίου 2-3-4:

Βάση των μετρικών η καλύτερη επιλογή για να επιτευχθεί η καλύτερη δυνατόν απόδοση ήταν ο αριθμός 3 συστάδων για τον L1 και 4 συστάδων για τους L2 και Max. Σε σύγκριση με το σενάριο 0 και 1, παρατηρούμε μια μεγάλη βελτίωση όσον αφορά την ποιότητα συσταδοποίησης που πραγματοποιείται.

Σε σύγκριση μεταξύ των διαφορετικών μεθόδων κανονικοποίησης δεδομένων, παρατηρείται με διαφορά καλύτερη ποιότητα συσταδοποίησης στην L1 (σκορ 1.21), ακολουθούμενη από την L2 (σκορ 1,07) και τελευταία την Max (σκορ 0.85). Από τις επιμέρους μετρικές, καθώς και από το τελικό σκορ συμπεραίνεται μια σχετικά καλή ποιότητα συσταδοποίησης. Πιο συμπυκνωμένα τα σημεία γύρω από τα κεντροειδή τους, με σχετικά καλά διαχωρισμένες συστάδες, σχεδόν χωρίς επικάλυψη και χωρίς μεγάλη διακύμανση εντός συστάδων. Η εφαρμογή μεθόδου κανονικοποίησης παρατηρείται ότι αποδίδει την καλύτερη ποιότητα συσταδοποίησης από όλα τα

σενάρια, όπως φαίνεται και στο Σχήμα 39. Η καλύτερη ποιότητα αποδίδεται στην L1, ακολουθούμενη από την L2 και Max.

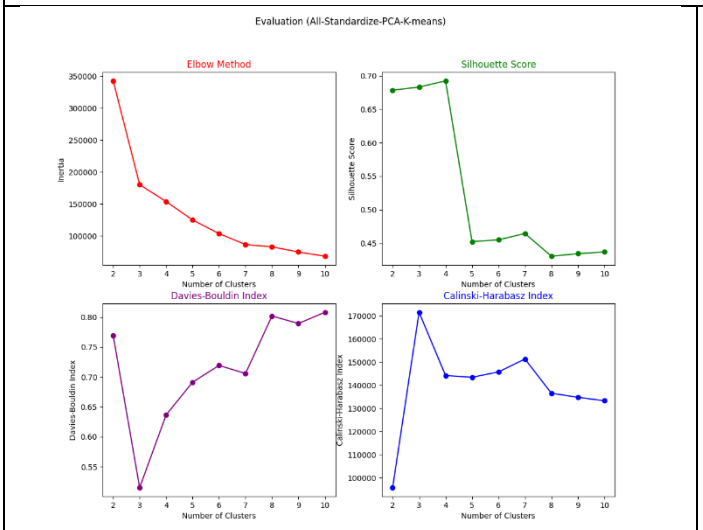
Κατά την οπτική αξιολόγηση της συσταδοποίησης, παρατηρείται καλύτερη συμπεριφορά της L1 όσον αφορά την διαχείριση των ακραίων σημείων, ακολουθούμενη από την L2. Η Max δεν φαίνεται να έχει διαχειριστεί αποτελεσματικά τα ακραία σημεία. Οι παρατηρήσεις αυτές επιβεβαιώνονται από το γεγονός ότι η κανονικοποίηση L1 δίνει έμφαση στη σπανιότητα και είναι ισχυρή σε ακραίες τιμές, η κανονικοποίηση L2 δίνει έμφαση στις μικρές διαφορές μεταξύ των παρατηρήσεων και είναι ευαίσθητη σε ακραίες τιμές και η κανονικοποίηση Max κλιμακώνει τα δεδομένα με βάση τη μέγιστη τιμή και διατηρεί την αρχική κλίμακα. Τέλος τα κέντρα των συστάδων βρίσκονται σε ικανοποιητική απόσταση.

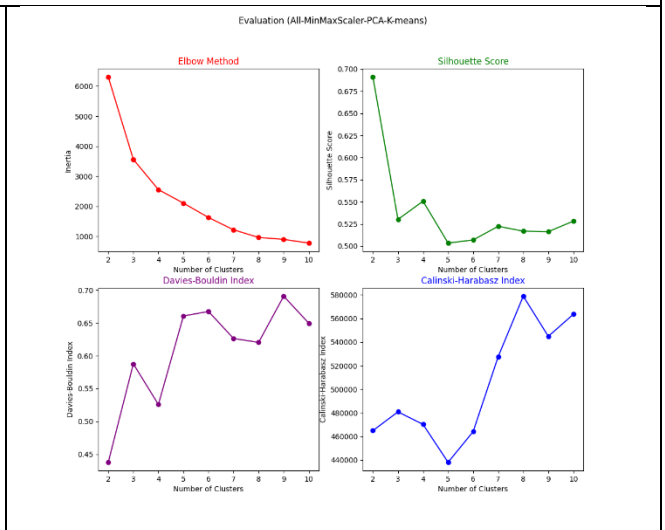
Παρατηρούμε στην οπτικοποίηση ότι κατά την κανονικοποίηση δεδομένων, ο διαχωρισμός των συστάδων δεν είναι ικανοποιητικός για την λήψη σωστών συμπερασμάτων, καθώς δεν μπορεί να αποτυπωθεί με ασφάλεια η διαφοροποίηση στην οδηγική συμπεριφορά. Φαίνεται σαν ο διαχωρισμός και εδώ να έχει γίνει με βάση μόνο των δεδομένων του επιταχυνσιόμετρου, διαχωρίζοντας τις συστάδες (3 για την L1 και 4 για L2, Max) βάση διαφορετικής ταχύτητας. Για παράδειγμα θα μπορούσε να συμπεράνει κάποιος για την L1 ότι η συστάδα 1 είναι φυσιολογική οδήγηση (μικρότερες τιμές κεντροειδούς), η συστάδα 2 είναι επιθετική ή επικίνδυνη οδήγηση με αυξημένη ταχύτητα και η συστάδα 0 είναι επιθετική ή επικίνδυνη οδήγηση με μεγάλη επιβράδυνση. Αντίστοιχα ο διαχωρισμός για την L2 θα μπορούσε να είναι φυσιολογική οδήγηση για την συστάδα 1 (μικρότερες τιμές κεντροειδούς), επικίνδυνη οδήγηση λόγω μεγάλης επιβράδυνσης για την συστάδα 2, επιθετική οδήγηση για την συστάδα 3 και επικίνδυνη οδήγηση λόγω αυξημένης ταχύτητας για την συστάδα 0. Ο διαχωρισμός για την Max θα μπορούσε να είναι φυσιολογική οδήγηση με επιτάχυνση για την συστάδα 3 (μικρότερες τιμές κεντροειδούς), φυσιολογική οδήγηση με επιβράδυνση για την συστάδα 1, επιθετική οδήγηση για την συστάδα 2 και επικίνδυνη οδήγηση λόγω αυξημένης ταχύτητας για την συστάδα 0.

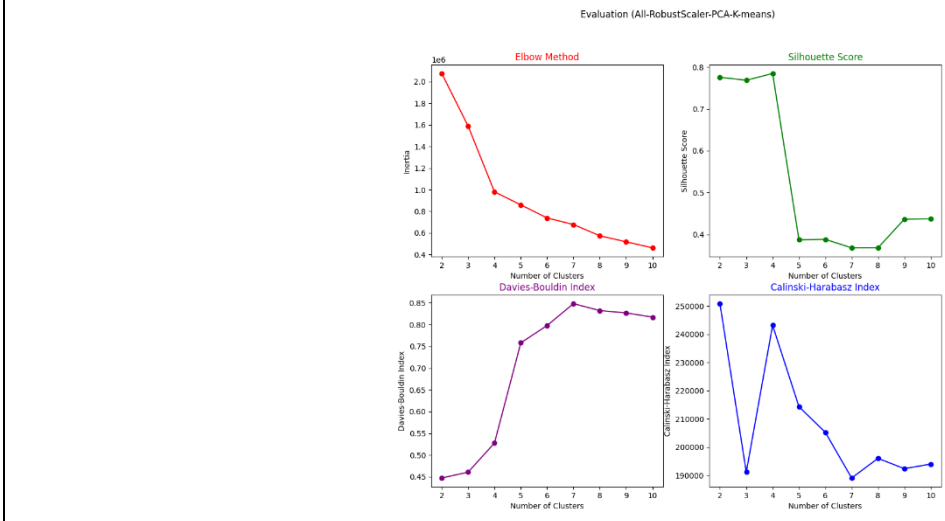
Βάση των παραπάνω, διακρίνεται σχετικά μεγάλη βελτίωση της ποιότητας της συσταδοποίησης χωρίς όμως να είναι αρκετή για ασφαλή συμπεράσματα. Από τα επόμενα σενάρια θα μπορέσει να βγει συμπέρασμα αν χρειάζεται συνδυαστικά η εφαρμογή μεθόδου κλιμάκωσης χαρακτηριστικών για απόδοση καλύτερων αποτελεσμάτων.

Στα επόμενα 3 σενάρια (s000100, s0000100, s0000001) γίνεται εφαρμογή των μεθόδων κλιμάκωσης χαρακτηριστικών (StandardScaler, MinMaxScaler, RobustScaler) και βάση των μετρικών αξιολόγησης ποιότητας συσταδοποίησης είχαμε τα εξής αποτελέσματα:

Σενάρια-5-6-7 (s000100, s0000100, s0000001)

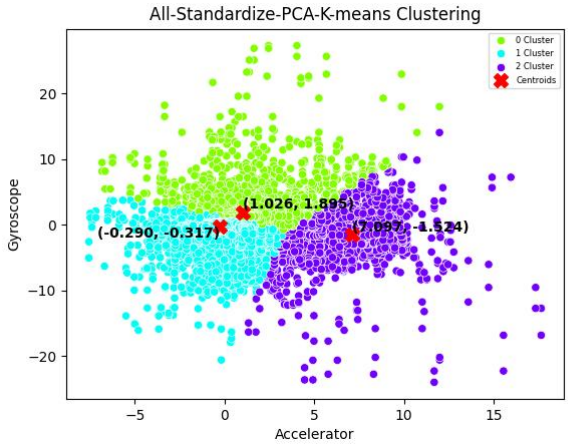


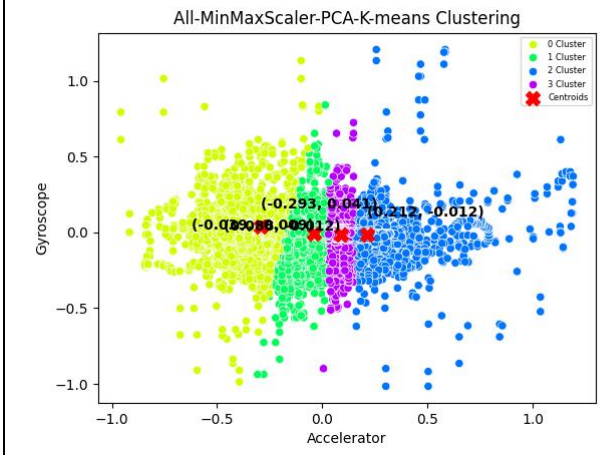


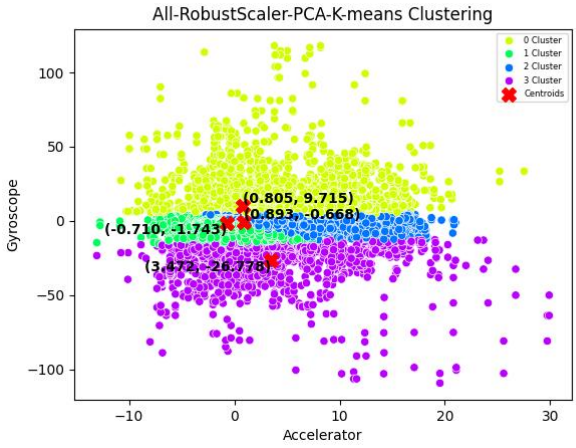


Inertia (Standard-MinMax-Robust):	Silhouette score (Standard-MinMax-Robust):	Davis-Bouldin Index (Standard-MinMax-Robust):	Calinski-Harabasz Index (Standard-MinMax-Robust):
180219.613 2552.817 980696.457	0.683 0.551 0.785	0.515 0.526 0.528	171452.489 470260.793 243161.606

Αρ. Συστάδων (Standard-MinMax-Robust):	Τελικό Σκορ (Standard-MinMax-Robust):	
3 4 4	0.5206942 0.699173028 0.041173886	Clusters Centers Values: Cluster 1 --> Acc: -0.290 -- Gyro: -0.317 Cluster 0 --> Acc: 1.026 -- Gyro: 1.895 Cluster 2 --> Acc: 7.097 -- Gyro: -1.524 Clusters Centers Values: Cluster 1 --> Acc: -0.039 -- Gyro: -0.009 Cluster 3 --> Acc: 0.088 -- Gyro: -0.012 Cluster 2 --> Acc: 0.212 -- Gyro: -0.012 Cluster 0 --> Acc: -0.293 -- Gyro: 0.041 Clusters Centers Values: Cluster 2 --> Acc: 0.893 -- Gyro: -0.668 Cluster 1 --> Acc: -0.710 -- Gyro: -1.743 Cluster 0 --> Acc: 0.805 -- Gyro: 9.715 Cluster 3 --> Acc: 3.472 -- Gyro: -26.778







<u>Παρατηρήσεις – Συμπεράσματα Σεναρίου 5-6-7:</u>

Βάση των μετρικών η καλύτερη επιλογή για να επιτευχθεί η καλύτερη δυνατόν απόδοση ήταν ο αριθμός 3 συστάδων για τον StandardScaler και 4 συστάδων για τους MinMaxScaler και RobustScaler.

Σε σύγκριση με το σενάριο 0 και 1, παρατηρούμε μια καλύτερη ποιότητα συσταδοποίησης για τον MinMaxScaler, για τον StandardScaler χειρότερη ποιότητα

από το σενάριο 1 (Winsorize) και καλύτερη από το σενάριο 0 (αυτούσια δεδομένα χωρίς εφαρμογή μεθόδου). Επίσης ο RobustScaler είναι χειρότερος ποιοτικά και από τα δύο σενάρια 0 και 1.

Στη σύγκριση των μεταξύ τους διαφορετικών μεθόδων κλιμάκωσης χαρακτηριστικών, παρατηρείται καλύτερη ποιότητα συσταδοποίησης στον MinMaxScaler (σκορ 0.70), ακολουθούμενος από τον StandardScaler (σκορ 0.52) και τελευταίος με διαφορά ο RobustScaler (σκορ 0.04). Από τις επιμέρους μετρικές, καθώς και από το τελικό σκορ συμπεραίνεται μια μέτρια ποιότητα συσταδοποίησης. Πιο συμπυκνωμένα τα σημεία γύρω από τα κεντροειδή τους για τον MinMaxScaler και πιο αραιά για τους άλλους δύο, με μικρότερη επικάλυψη για τον RobustScaler και με καλύτερα διαχωρισμένες συστάδες χωρίς μεγάλη διακύμανση εντός συστάδων για τον StandardScaler. Η εφαρμογή μεθόδου κλιμάκωσης χαρακτηριστικών παρατηρείται ότι αποδίδει μέτρια αλλά σχετικά καλύτερα από τα περισσότερα σενάρια κατέχοντας την $4^{η}$ θέση ο MinMaxScaler, την $6^{η}$ θέση ο StandardScaler και την $9^{η}$ θέση ο RobustScaler, όπως φαίνεται και στο Σχήμα 39. Η καλύτερη ποιότητα συσταδοποίησης στην μεταξύ τους σχέση αποδίδεται στον MinMaxScaler, ακολουθούμενος από τον StandardScaler και με μεγαλύτερη διαφορά από τον RobustScaler.

Κατά την οπτική αξιολόγηση της συσταδοποίησης, παρατηρείται εμφάνιση ακραίων σημείων και στις τρεις περιπτώσεις. Η κλιμάκωση των ακραίων σημείων φαίνεται καλύτερη στον MinMaxScaler. Η κλιμάκωση των χαρακτηριστικών φαίνεται κατανοητή και στις τρεις περιπτώσεις. Τα κέντρα των συστάδων δεν βρίσκονται σε ικανοποιητική απόσταση.

Παρατηρούμε στην οπτικοποίηση ότι κατά την κλιμάκωση χαρακτηριστικών, ο διαχωρισμός των συστάδων δεν είναι ικανοποιητικός για την λήψη σωστών συμπερασμάτων ειδικά για τον MinMaxScaler, καθώς δεν μπορεί να αποτυπωθεί με ασφάλεια η διαφοροποίηση στην οδηγική συμπεριφορά. Στην περίπτωση του MinMaxScaler ο διαχωρισμός φαίνεται να έχει γίνει με βάση μόνο των δεδομένων του επιταχυνσιόμετρου, διαχωρίζοντας τις 4 συστάδες βάση διαφορετικής ταχύτητας. Για παράδειγμα θα μπορούσε να συμπεράνει κάποιος για την MinMaxScaler ότι η συστάδα 1 θα μπορούσε να είναι φυσιολογική οδήγηση με επιβράδυνση (μικρότερες τιμές κεντροειδούς), φυσιολογική οδήγηση με επιτάχυνση για την συστάδα 3, επιθετική ή επικίνδυνη οδήγηση λόγω αυξημένης ταχύτητας για την συστάδα 2 και επιθετική ή επικίνδυνη οδήγηση λόγω αυξημένης επιβράδυνσης ταχύτητας για την

συστάδα 0. Στην περίπτωση του StandardScaler ο διαχωρισμός φαίνεται να έχει λάβει υπόψη τα δεδομένα και των δύο αισθητήρων, διαχωρίζοντας τις 3 συστάδες βάση διαφορετικής οδηγικής συμπεριφοράς. Για παράδειγμα θα μπορούσε να συμπεράνει κάποιος για την StandardScaler ότι η συστάδα 1 θα μπορούσε να είναι φυσιολογική οδήγηση (μικρότερες τιμές κεντροειδούς), επιθετική οδήγηση για την συστάδα 0, επικίνδυνη οδήγηση για την συστάδα 2. Το παράλογο στην ερμηνεία που κάνει τον διαχωρισμό συστάδων μη ικανοποιητικό για ασφαλή συμπεράσματα είναι ότι π.χ. οι μεγάλες αρνητικές τιμές του γυροσκοπίου στην συστάδα 1 να θεωρούνται φυσιολογική οδήγηση. Στην περίπτωση του RobustScaler ο διαχωρισμός φαίνεται να έχει λάβει υπόψη τα δεδομένα και των δύο αισθητήρων αλλά δίνοντας έμφαση στις στροφές δηλαδή στα δεδομένα του γυροσκοπίου, διαχωρίζοντας τις 4 συστάδες βάση διαφορετικής οδηγικής συμπεριφοράς. Για παράδειγμα θα μπορούσε να συμπεράνει κάποιος για την RobustScaler ότι η συστάδα 2 θα μπορούσε να είναι φυσιολογική οδήγηση χωρίς ακραίες στροφές με τάση στην επιτάχυνση (μικρότερες τιμές κεντροειδούς), φυσιολογική οδήγηση χωρίς ακραίες στροφές με τάση στην επιβράδυνση για την συστάδα 1, επιθετική ή επικίνδυνη οδήγηση με ακραίες στροφές προς την μια κατεύθυνση για την συστάδα 0 και επιθετική ή επικίνδυνη οδήγηση με ακραίες στροφές προς την άλλη κατεύθυνση για την συστάδα 3.

Βάση των παραπάνω, διακρίνεται μέτρια ποιότητα της συσταδοποίησης χωρίς όμως να είναι αρκετή για ασφαλή συμπεράσματα. Η κάθε μια από τις μεθόδους έχει πλεονεκτήματα και μειονεκτήματα. Από μόνες τους αυτές οι μέθοδοι δεν είναι αρκετές για τα συμπεράσματα που απαιτεί η εφαρμογή.

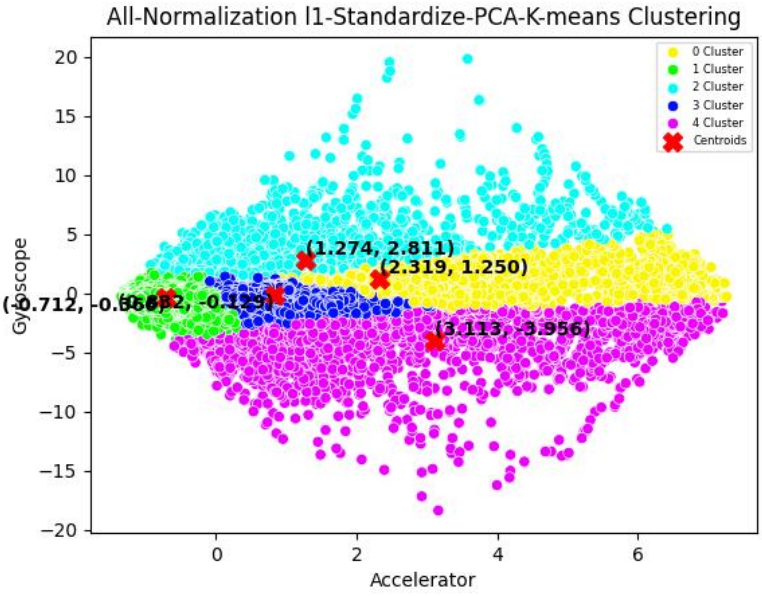
Συμπεράσματα Εκτέλεσης Σεναρίων

Βάση των μετρικών αξιολόγησης ποιότητας συσταδοποίησης (Inertia, Silhouette score, Calinski-Harabasz index, Davies-Bouldin index), στα σενάρια εφαρμογής μεθόδων διαχείρισης ακραίων σημείων, κανονικοποίησης δεδομένων και κλιμάκωσης χαρακτηριστικών, καθώς και συνδυασμών αυτών, με σειρά συνδυασμού

Διαχ. Ακραίων Σημείων → Κανονικοποίηση → Κλιμάκωση Χαρακτηριστικών

καταλήγουμε στα εξής συμπεράσματα, που οπτικοποιούνται και στο διάγραμμα του σχήματος 39:

- Την καλύτερη απόδοση παρουσιάζουν οι εφαρμογές των τριών μεθόδων κανονικοποίησης με καλύτερη την L1, μετά την L2 και την Max
- Την δεύτερη καλύτερη απόδοση παρουσιάζουν οι εφαρμογές μεθόδων κλιμάκωσης χαρακτηριστικών με καλύτερη την MinMaxScaler, μετά την StandardScaler και τελευταία την RobustScaler.
- Στις χειρότερες περιπτώσεις σεναρίων κατατάσσεται ο συνδυασμός Winsorize (διαχείριση ακραίων σημείων) με StandardScaler (κλιμάκωση χαρακτηριστικών), ακολουθούμενη από τους συνδυασμούς Winsorize-L2-RobustScaler και L2-RobustScaler.
- Όλα τα υπόλοιπα σενάρια συνδυασμών διαχείρισης ακραίων σημείων, κανονικοποίησης και κλιμάκωσης χαρακτηριστικών, δεν αποδίδουν καθόλου δημιουργώντας μια κακή ποιότητα συσταδοποίησης.
- Η εφαρμογή διαχείρισης ακραίων σημείων Winsorize όταν συμμετέχει σε οποιοδήποτε συνδυασμό δεν αποδίδει καλά. Εξαίρεση αποτελεί ο συνδυασμός Winsorize-L1-RobustScaler που αποδίδει μέτρια αλλά με 8 συστάδες, γεγονός το οποίο έχει ως αποτέλεσμα την υπέρ-ανάλυση της πληροφορίας που δεν εξυπηρετεί την εφαρμογή μας.
- Ο συνδυασμός μεθόδου κανονικοποίησης ακολουθούμενη συνδυαστικά με μέθοδο κλιμάκωσης χαρακτηριστικών, δεν αποδίδει καλά.
- Στην αξιολόγηση της συσταδοποίησης βάση οπτικής παρατήρησης, συμπεραίνουμε ότι κανένα σενάριο δεν κατέστη ικανό να αποδώσει κατάλληλη πληροφορία για λήψη σωστών συμπερασμάτων της οδηγικής συμπεριφοράς που είναι και το ζητούμενο της εφαρμογής μας. Ενδιαφέρον παρουσιάζει ωστόσο η οπτικοποίηση του συνδυασμού L1-StandardScaler που ενώ έχει κακή ποιότητα συσταδοποίησης βάση μετρικών (σκορ -0.79), έχει καταμερισμό συστάδων για μια λογική αποτύπωση διαχωρισμού της οδηγικής συμπεριφοράς.



Clusters Centers Values:
Cluster 3 --> Acc: 0.832 -- Gyro: -0.129
Cluster 1 --> Acc: -0.712 -- Gyro: -0.366
Cluster 0 --> Acc: 2.319 -- Gyro: 1.250
Cluster 2 --> Acc: 1.274 -- Gyro: 2.811
Cluster 4 --> Acc: 3.113 -- Gyro: -3.956

Σχήμα 40: Συσταδοποίηση συνδυασμού L1-StandardScaler

όπου θα μπορούσε κάποιος να χαρακτηρίσει την συστάδα 3 ως φυσιολογική οδήγηση με έμφαση στην επιτάχυνση της ταχύτητας, την συστάδα 1 ως φυσιολογική οδήγηση με έμφαση στην επιβράδυνση της ταχύτητας, την συστάδα 0 ως επιθετική ή επικίνδυνη οδήγηση με έμφαση στην επιτάχυνση της ταχύτητας και τις συστάδες 2 και 4 ως επιθετική ή επικίνδυνη οδήγηση με έμφαση στις απότομες στροφές ανεξαρτήτως ταχύτητας.

Γενικό συμπέρασμα είναι ότι ο συνδυασμός των μεθόδων μετασχηματισμού με την συγκεκριμένη σειρά έχει μέτρια ως κακή απόδοση και η χρήση του αλγόριθμου K-means με αυτά τα εργαλεία δεν είναι ικανή να μας παρέχει σωστά και ερμηνεύσιμα συμπεράσματα οδηγικής συμπεριφοράς.

Εκτέλεση 2ης Ομάδας Σεναρίων

Λαμβάνοντας υπόψη τα παραπάνω συμπεράσματα, αποφασίστηκε η εκτέλεση επιπλέον σεναρίων τα οποία θα δίνουν έμφαση στην σειρά εφαρμογής μεθόδων μετασχηματισμού δεδομένων ως εξής:

Διαχ. Ακραίων Σημείων → Κλιμάκωση Χαρακτηριστικών → Κανονικοποίηση

Τα σενάρια αυτά είναι τα εξής:

No	Senario	DATA	ODR	FS	ND	PCA	ULA
0	0100100	All	-	StandarScaler	L1	√	K-means
1	0100010	All	-	StandarScaler	L2	√	K-means
2	0100001	All	-	StandarScaler	Max	√	K-means
3	0010100	All	-	MinMaxScaler	L1	√	K-means
4	0010010	All	-	MinMaxScaler	L2	√	K-means
5	0010001	All	-	MinMaxScaler	Max	√	K-means
6	0001100	All	-	RobustScaler	L1	√	K-means
7	0001010	All	-	RobustScaler	L2	√	K-means
8	0001001	All	-	RobustScaler	Max	√	K-means
9	1100100	All	Winsorize	StandarScaler	L1	√	K-means
10	1100010	All	Winsorize	StandarScaler	L2	√	K-means
11	1100001	All	Winsorize	StandarScaler	Max	√	K-means
12	1010100	All	Winsorize	MinMaxScaler	L1	√	K-means
13	1010010	All	Winsorize	MinMaxScaler	L2	√	K-means
14	1010001	All	Winsorize	MinMaxScaler	Max	√	K-means
15	1001100	All	Winsorize	RobustScaler	L1	√	K-means
16	1001010	All	Winsorize	RobustScaler	L2	√	K-means
17	1001001	All	Winsorize	RobustScaler	Max	√	K-means

Η σειρά εκτέλεσης είναι:

Winsorize →StandardScaler→MinMaxScaler→RobustScaler→L1→L2→Max

(π.χ. το scenario 17 1001001 έχει άσσο στην 1η 4η και 7η θέση άρα θα εκτελέσει Winsorize →RobustScaler→Max).

Η κωδικοποίηση των σεναρίων βρίσκεται σε αρχείο με όνομα ***executionSenarios2.csv*** και βρίσκεται στην τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

Το αρχείο αυτό φορτώνεται και χρησιμοποιείται μέσα από τον κώδικα εκτέλεσης των σεναρίων.

Αφού εκτελέστηκε η 2η ομάδα σεναρίων, τα συγκεντρωτικά αποτελέσματα όλων των σεναρίων είναι τα εξής:

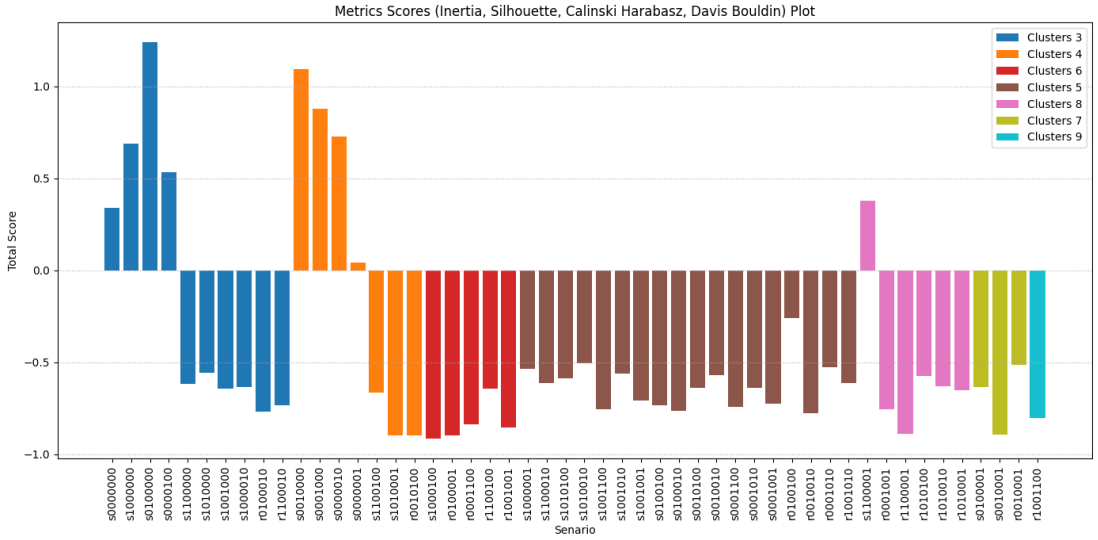
Senario	Title	Clusters	Inertia	Silhouette	Calinski	Davies	TotalScore
s0000000	All-PCA-K-means	3	274274,703	0,551	500836,874	0,571	0,337149218
s1000000	All-Winsorizer-PCA-K-means	3	207788,373	0,561	642145,892	0,552	0,689095535
s0100000	All-Normalization l1-PCA-K-means	3	842,444	0,604	813584,832	0,555	1,237887498
s0010000	All-Normalization l2-PCA-K-means	4	1396,397	0,568	669119,985	0,512	1,093034472
s0001000	All-Normalization max-PCA-K-means	4	2224,419	0,558	538978,801	0,515	0,878424349
s0000100	All-Standardize-PCA-K-means	3	180219,613	0,683	171452,489	0,515	0,533131529
s0000010	All-MinMaxScaler-PCA-K-means	4	2552,817	0,551	470260,793	0,526	0,728136313
s0000001	All-RobustScaler-PCA-K-means	4	980696,457	0,785	243161,606	0,528	0,042141823
s1100000	All-Winsorizer-Normalization l1-PCA-K-means	3	9941,914	0,48	202473,176	0,753	-0,615918189
s1010000	All-Winsorizer-Normalization l2-PCA-K-means	3	26599,916	0,49	221521,27	0,746	-0,556121518
s1001000	All-Winsorizer-Normalization max-PCA-K-means	3	46192,649	0,486	215403,518	0,76	-0,643419017
s1000100	All-Winsorizer-Standardize-PCA-K-means	6	75667,943	0,431	183956,974	0,775	-0,915352688
s1000010	All-Winsorizer-MinMaxScaler-PCA-K-means	3	46219,53	0,487	215462,235	0,759	-0,637237527
s1000001	All-Winsorizer-RobustScaler-PCA-K-means	5	68307,985	0,471	190148,868	0,701	-0,537027559
s1100100	All-Winsorizer-Normalization l1-Standardize-PCA-K-means	4	114654,895	0,464	176689,068	0,714	-0,666187593
s1100010	All-Winsorizer-Normalization l1-MinMaxScaler-PCA-K-means	5	6958,688	0,45	213215,853	0,734	-0,613206784
s1100001	All-Winsorizer-Normalization l1-RobustScaler-PCA-K-means	8	62988,697	0,428	908334,565	0,686	0,37947145
s1010100	All-Winsorizer-Normalization l2-Standardize-PCA-K-means	5	90977,578	0,473	224975,068	0,724	-0,586850093
s1010010	All-Winsorizer-Normalization l2-MinMaxScaler-PCA-K-means	5	16536,69	0,471	234906,114	0,725	-0,506211775
s1010001	All-Winsorizer-Normalization l2-RobustScaler-PCA-K-means	4	74195,539	0,454	206667,164	0,797	-0,897230581
s1001100	All-Winsorizer-Normalization max-	5	101148,687	0,453	205340,839	0,747	-0,756804978

	Standardize-PCA-K-means						
s1001010	All-Winsorizer-Normalization max-MinMaxScaler-PCA-K-means	5	30430,893	0,462	223674,352	0,726	-0,563210886
s1001001	All-Winsorizer-Normalization max-RobustScaler-PCA-K-means	5	60096,652	0,447	210093,446	0,742	-0,7073037
s0100100	All-Normalization l1-Standardize-PCA-K-means	5	92270,084	0,453	208702,989	0,745	-0,736284607
s0100010	All-Normalization l1-MinMaxScaler-PCA-K-means	5	10186,809	0,438	214110,095	0,768	-0,764700689
s0100001	All-Normalization l1-RobustScaler-PCA-K-means	7	53299,196	0,467	189659,318	0,731	-0,636546676
s0010100	All-Normalization l2-Standardize-PCA-K-means	5	90348,596	0,466	225424,944	0,734	-0,638847181
s0010010	All-Normalization l2-MinMaxScaler-PCA-K-means	5	16536,135	0,464	232699,018	0,738	-0,572800733
s0010001	All-Normalization l2-RobustScaler-PCA-K-means	7	41971,14	0,443	202586,694	0,795	-0,892475187
s0001100	All-Normalization max-Standardize-PCA-K-means	5	99659,486	0,45	207727,226	0,742	-0,742848182
s0001010	All-Normalization max-MinMaxScaler-PCA-K-means	5	30382,495	0,45	220805,724	0,738	-0,640605312
s0001001	All-Normalization max-RobustScaler-PCA-K-means	5	61088,499	0,445	206366,97	0,744	-0,725631507
r0100100	All-Standardize-Normalization l1-PCA-K-means	5	3200,631	0,47	351286,309	0,703	-0,261484437
r0100010	All-Standardize-Normalization l2-PCA-K-means	3	22360,78	0,472	238330,551	0,802	-0,770344687
r0100001	All-Standardize-Normalization max-PCA-K-means	6	23490,705	0,412	212981,594	0,782	-0,897790525
r0010100	All-MinMaxScaler-Normalization l1-PCA-K-means	4	5263,154	0,432	196823,647	0,797	-0,8992205
r0010010	All-MinMaxScaler-Normalization l2-PCA-K-means	5	13395,024	0,435	216898,177	0,77	-0,779130692
r0010001	All-MinMaxScaler-Normalization max-PCA-K-means	7	18397,302	0,456	235559,026	0,715	-0,512956313
r0001100	All-RobustScaler-Normalization l1-PCA-K-means	6	4040,441	0,439	186756,999	0,781	-0,837694617
r0001010	All-RobustScaler-Normalization l2-PCA-K-means	5	12774,743	0,47	238398,809	0,733	-0,527899825
r0001001	All-RobustScaler-Normalization max-PCA-K-means	8	17306,518	0,428	212102,132	0,755	-0,75667379
r1100100	All-Winsorizer-Standardize-Normalization l1-PCA-K-means	6	2610,392	0,439	290621,773	0,766	-0,643559337
r1100010	All-Winsorizer-Standardize-Normalization l2-PCA-K-means	3	22083,019	0,475	242221,72	0,796	-0,736048079

r1100001	All-Winsorizer-Standardize-Normalization max-PCA-K-means	8	17509,25	0,422	217455,738	0,791	-0,889839199
r1010100	All-Winsorizer-MinMaxScaler-Normalization l1-PCA-K-means	8	2250,687	0,447	229910,196	0,729	-0,576548123
r1010010	All-Winsorizer-MinMaxScaler-Normalization l2-PCA-K-means	8	8074,303	0,447	252810,85	0,752	-0,630724045
r1010001	All-Winsorizer-MinMaxScaler-Normalization max-PCA-K-means	8	16292,587	0,454	250541,331	0,76	-0,651010644
r1001100	All-Winsorizer-RobustScaler-Normalization l1-PCA-K-means	9	2966,268	0,436	172471,142	0,763	-0,801959185

Με ‘s’ συμβολίζονται τα σενάρια της 1ης ομάδας σεναρίων και με ‘r’ τα σενάρια της 2ης ομάδας.

Οπτικοποιήσαμε τα συνολικά αποτελέσματα μέσω γραφήματος μπάρας, αναπαριστώντας το κάθε σενάριο σε σχέση με το τελικό σκορ που επιτεύχθηκε. Με χρώμα και ετικέτες αναπαρίστανται οι βέλτιστοι αριθμοί συστάδων για το κάθε σενάριο. Στο γράφημα αποτυπώνεται η ποιότητα συσταδοποίησης του κάθε σεναρίου.



Σχήμα 41: Συγκεντρωτικό γράφημα ποιότητας συσταδοποίησης σεναρίων

Οι μετρικές των σεναρίων της 2ης ομάδας προστέθηκαν στις συνολικές μετρικές, αποδίδοντας την τελική εικόνα.

Συμπεράσματα Εκτέλεσης Σεναρίων

Βάση των μετρικών αξιολόγησης ποιότητας συσταδοποίησης (Inertia, Silhouette score, Calinski-Harabasz index, Davies-Bouldin index), στην $2^{η}$ ομάδα σεναρίων εφαρμογής μεθόδων διαχείρισης ακραίων σημείων, κανονικοποίησης δεδομένων και κλιμάκωσης χαρακτηριστικών, καθώς και συνδυασμών αυτών, με σειρά συνδυασμού

Διαχ. Ακραίων Σημείων → Κλιμάκωση Χαρακτηριστικών → Κανονικοποίηση

καταλήγουμε στα εξής συμπεράσματα, που οπτικοποιούνται και στο διάγραμμα του σχήματος 41:

- Η εκτέλεση των σεναρίων της $2^{ης}$ ομάδας σεναρίων, δεν συνεισφέραν καθόλου στην βελτίωση της ποιότητας συσταδοποίησης αφού δεν είχαν καθόλου καλό αποτέλεσμα επιμέρους μετρικών και τελικού σκορ.
- Τα σενάρια της $1^{ης}$ ομάδας σεναρίων παραμένουν στην ίδια ιεράρχηση σκορ.
- Η αντίστροφη εφαρμογή μεθόδων από κανονικοποίηση-κλιμάκωση χαρακτηριστικών σε κλιμάκωση χαρακτηριστικών-κλιμάκωση, δεν είχε θετικό αποτέλεσμα και δεν συνετέλεσε σημαντικά στην ποιότητα συσταδοποίησης.
- Οι καλύτερες αποδόσεις παραμένουν αυτές που αναφέρθηκαν στα συμπεράσματα της $1^{ης}$ ομάδας σεναρίων.
- Οι χειρότερες αποδόσεις παραμένουν αυτές που αναφέρθηκαν στα συμπεράσματα της $1^{ης}$ ομάδας σεναρίων.

Γενικά Συμπεράσματα

Πραγματοποιήθηκε η εφαρμογή του αλγόριθμου K-means στο σύνολο δεδομένων μας, διενεργώντας πολλαπλούς συνδυασμούς διαφορετικών μεθόδων διαχείρισης ακραίων σημείων, κανονικοποίησης δεδομένων και κλιμάκωσης χαρακτηριστικών.

Βάση αυτών των εφαρμογών ανιχνεύτηκε ότι για το σύνολο των δεδομένων μας καλύτερα και πιο ποιοτικά αποτελέσματα συσταδοποίησης στην εφαρμογή του K-means παρείχε η προγενέστερη εφαρμογή μεθόδου κανονικοποίησης δεδομένων (L1,

L2, Max) με πιο αποδοτική αυτή της κανονικοποίησης δεδομένων L1. Το γεγονός αυτό είναι λογικό λαμβάνοντας υπόψη ότι ο L1 είναι ανθεκτικός σε ακραίες τιμές (δεν βασίζεται σε τετραγωνικούς όρους) και δίνει έμφαση στην σπανιότητα παρατηρήσεων. Υπερτερεί της L2 που είναι ευαίσθητη σε ακραίες τιμές αλλά δίνει έμφαση στις μικρές διαφορές μεταξύ παρατηρήσεων γεγονός που την κατατάσσει στις καλύτερες αποδόσεις.

Τα αμέσως επόμενα καλύτερα σε ποιότητα συσταδοποίησης αποτελέσματα τα κατέχει η εφαρμογή κλιμάκωσης χαρακτηριστικών (StandardScaler, MinMaxScaler, RobustScaler), με πιο αποδοτική αυτή της κλιμάκωσης χαρακτηριστικών με την MinMaxScaler. Ο MinMaxScaler δεν λαμβάνει υπόψη την κατανομή των δεδομένων και παραμετροποιώντας τον στο εύρος (-1, +1) μετριάσαμε τον αντίκτοιπο των ακραίων τιμών, στοιχεία που βοήθησαν στην κατάταξή του στις καλύτερες αποδόσεις. Υπερτέρησε της StandardScaler που είναι ευαίσθητη σε ακραίες τιμές και της RobustScaler που ενώ είναι ανθεκτική στις ακραίες τιμές και λοξότητα την δεσμεύει η κανονική κατανομή των δεδομένων.

Το γενικό συμπέρασμα, είναι ότι ο αλγόριθμος K-means εξήγαγε μέτριας ποιότητα συσταδοποίηση για την λήψη λογικών και ερμηνεύσιμων συμπερασμάτων διαχωρισμού συμπεριφορών οδήγησης. Οι συνδυασμοί διαφορετικών μεθόδων διαχείρισης ακραίων σημείων, κανονικοποίησης δεδομένων και κλιμάκωσης χαρακτηριστικών, τροφοδότησαν τον αλγόριθμο συσταδοποίησης K-means με δεδομένα, τέτοια ώστε η λειτουργία του να είχε μέτρια απόδοση.
Τα συμπεράσματα αυτά είναι λογικά, αν λάβει κανείς υπόψη ότι ο αλγόριθμος K-means είναι ευαίσθητος σε ακραία σημεία, δεν είναι εύκολο να διαχωρίσει δεδομένα με διαφορετικές πυκνότητες και τείνει να δημιουργεί σφαιρικές συστάδες (λόγω υπολογισμού συστάδων βάση απόστασης από τα κεντροειδή τους). Όλα αυτά, είναι χαρακτηριστικά του συνόλου των δεδομένων μας και για τον λόγο αυτό, ο αλγόριθμος K-means, κρίνεται μέτριας απόδοσης για την υλοποίηση της εφαρμογής μας.

Πιο κατάλληλοι πιθανότατα, να είναι αλγόριθμοι συσταδοποίησης που να βασίζονται στην πυκνότητα, όπως είναι ο DBSCAN ή παραλλαγές αυτού (π.χ. OPTICS,

HDBSCAN). Οι αλγόριθμοι αυτοί είναι ανθεκτικοί στα μειονεκτήματα του K-means που προαναφέρθηκαν προηγουμένως, καθιστώντας τους καλύτερους υποψήφιους για το σύνολο δεδομένων μας και την εφαρμογή πρόβλεψης συμπεριφοράς οδήγησης.

Ο αλγόριθμος DBSCAN (Density-Based Spatial Clustering of Applications with Noise), βασίζεται στην έννοια της πυκνότητας. Ορίζει συστάδες ως συνεχείς περιοχές υψηλής πυκνότητας, χωρίς να χρειάζεται να γνωρίζουμε τον αριθμό των συστάδων εκ των προτέρων. Ανιχνεύει εύκολα συστάδες διάφορων μορφών και μεγεθών. Μπορεί να αντιμετωπίσει το θόρυβο στα δεδομένα, ανιχνεύοντας συστάδες ακραίων σημείων (noise) σε δεδομένα. Επειδή βασίζεται στην πυκνότητα, ο DBSCAN μπορεί να αναγνωρίσει συνεχείς συστάδες χωρίς να επηρεάζεται από ασυνεχή ή κατακερματισμένα σύνολα δεδομένων. Στα μειονεκτήματά του, είναι η ευαισθησία που έχει στην δήλωση των παραμέτρων του που πρέπει να ρυθμιστούν, όπως ο ελάχιστος αριθμός γειτόνων και η ελάχιστη απόσταση, καθώς η επιλογή αυτών μπορεί να επηρεάσει σημαντικά τα αποτελέσματα. Επίσης, δυσκολεύεται να αναγνωρίσει συστάδες με διαφορετικές πυκνότητες, καθώς οι παράμετροι που επιλέγονται για ένα σύνολο δεδομένων μπορεί να μην είναι κατάλληλες για άλλα τμήματα των δεδομένων. Η χρονική πολυπλοκότητα του αλγόριθμου, είναι συνήθως εκθετική καθιστώντας τον συνήθως αποδοτικό για μεσαία σύνολα δεδομένων, καθώς μπορεί να γίνει απαιτητικός σε υπολογιστικούς πόρους για μεγάλα ή υψηλών διαστάσεων δεδομένα.

Ο αλγόριθμος OPTICS (Ordering Points To Identify the Clustering Structure), είναι παρόμοιος με τον DBSCAN, αλλά δεν απαιτεί την προκαθορισμένη επιλογή των παραμέτρων. Αναπτύχθηκε για να αντιμετωπίσει το πρόβλημα των πολλών διαστάσεων που παρουσιάζει ο DBSCAN. Προσφέρει μια γενική μέθοδο συσταδοποίησης χωρίς την ανάγκη να οριστούν εκ των προτέρων παράμετροι. Μπορεί να ανιχνεύσει συστάδες με διαφορετικές πυκνότητες, χωρίς να απαιτείται η προσδιορισμός ενός σταθερού κατωφλίου πυκνότητας. Είναι ανθεκτικός στο θόρυβο και τις ακραίες περιοχές, καθώς μπορεί να τα αντιμετωπίσει ως απλή απόκλιση από την κύρια δομή των συστάδων. Στα μειονεκτήματά του είναι ότι μπορεί να απαιτεί αρκετούς υπολογιστικούς πόρους, ειδικά για μεγάλα σύνολα δεδομένων, λόγω του υπολογισμού των αποστάσεων μεταξύ όλων των σημείων. Επειδή ο αλγόριθμος OPTICS απαιτεί την ανάλυση όλων των γειτονικών σημείων για κάθε δείγμα, η

χρονική πολυπλοκότητά του μπορεί να αυξηθεί σημαντικά σε σύγκριση με τον αλγόριθμο DBSCAN, ειδικά για μεγάλα σύνολα δεδομένων.

Ο αλγόριθμος HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), είναι μια βελτίωση του DBSCAN που αυτοματοποιεί τη διαδικασία εύρεσης των βέλτιστων παραμέτρων. Βασίζεται στην ιδέα της ιεραρχικής συσταδοποίησης, χρησιμοποιώντας ένα δέντρο βασισμένο στην πυκνότητα. Δεν απαιτεί τον προκαθορισμό των αριθμών των συστάδων πριν την εκτέλεση του αλγορίθμου. Παρέχει τη δυνατότητα να ανιχνεύσει συστάδες διαφορετικών μεγεθών και πυκνοτήτων. Απαιτείται λιγότερος χρόνος για τη ρύθμιση των παραμέτρων συγκριτικά με τον DBSCAN. Μπορεί να αντιμετωπίσει τον θόρυβο ως σύνολο απλών σημείων που δεν ανήκουν σε καμία συστάδα. Μπορεί να ανιχνεύσει συστάδες με διαφορετικές πυκνότητες, προσαρμόζοντας αυτόματα το επίπεδο της πυκνότητας για κάθε συστάδα. Με την κατάλληλη παραμετροποίηση, παράγει συστάδες υψηλής ποιότητας, λαμβάνοντας υπόψη την πυκνότητα των δεδομένων και την δομή των συστάδων. Παρά την ικανότητά του να ανιχνεύει συστάδες διαφορετικών μεγεθών και πυκνοτήτων, είναι συνήθως πολύ αποδοτικός, ειδικά σε μεγάλα σύνολα δεδομένων. Στα μειονεκτήματά του, μπορεί να απαιτεί σημαντικούς υπολογιστικούς πόρους, ειδικά για μεγάλα σύνολα δεδομένων ή δεδομένα υψηλών διαστάσεων. Απαιτεί την επιλογή ενός συνόλου παραμέτρων, όπως ο παράμετρος min_cluster_size, η οποία μπορεί να είναι δύσκολο να προσδιοριστεί χωρίς προηγούμενη γνώση των δεδομένων ή δοκιμής και σφάλματος και της παραμέτρου min_samples που μπορεί να επηρεάσει σημαντικά τα αποτελέσματα του αλγορίθμου, καθώς μικρές αλλαγές σε αυτή την παράμετρο μπορεί να οδηγήσουν σε διαφορετικές συστάδες. Γενικά, η χρονική πολυπλοκότητα του HDBSCAN είναι γραμμική και καλύτερη από τον αλγόριθμο DBSCAN. Είναι συνήθως αποδεκτή για μεγάλα σύνολα δεδομένων, καθιστώντας τον αλγόριθμο αποτελεσματικό για τη συσταδοποίηση σε πραγματικές εφαρμογές.

Λαμβάνοντας υπόψη τις παραπάνω συγκρίσεις στα πλεονεκτήματα και τα μειονεκτήματα των υποψηφίων αλγορίθμων καθώς και της χρονικής πολυπλοκότητας που απαιτούν, ίσως η καλύτερη λύση να είναι η εφαρμογή του αλγόριθμου HDBSCAN.

Κώδικας και Αρχεία Εφαρμογής

Τα στάδια εκτέλεσης της εφαρμογής έχουν ως εξής:

1. Εκτελείται ο κώδικας προ-επεξεργασίας και μετασχηματισμού του αρχικού συνόλου δεδομένων (από το ***data.zip***) με όνομα ***preprocessData.py***. Αποτέλεσμα είναι το τελικό σύνολο δεδομένων με όνομα ***finalData.csv.zip***
2. Εκτελείται ο κώδικας διερευνητικής ανάλυσης δεδομένων στο αρχείο ***finalData.csv.zip*** που περιέχει το τελικό σύνολο των δεδομένων, με όνομα ***datasetEvaluation.py***.
3. Εκτελείται ο κώδικας ανίχνευσης καλύτερης ποιότητας συσταδοποίησης βάση σεναρίων στο αρχείο ***finalData.csv.zip*** με όνομα ***DriverPredictSenarios_(K-means).py***
4. Εκτελείται ο κώδικας οπτικοποίησης συσταδοποίησης για συγκεκριμένο σενάριο στο αρχείο ***finalData.csv.zip*** με όνομα ***DriverPredict_(K-means).py***

Όλα τα αρχεία κώδικα και δεδομένων μπορούν να αντληθούν από την τοποθεσία: https://drive.google.com/drive/folders/1QAbfHxqyHjUZXWYzl4ZHHrQO276CmYtH?usp=drive_link

Τα αρχεία δεδομένων (***data.zip, finalData.csv.zip, executionSenarios.csv, executionSenarios2.csv, clustersMetrics.csv, clustersMetrics.xlsx***), πρέπει να βρίσκονται τοπικά στην τοποθεσία *C:\datasets*

Το αρχείο ***data.zip*** αφού τοποθετηθεί στον παραπάνω φάκελο, θα πρέπει να αποσυμπιεστεί, δημιουργώντας τον φάκελο *C:\datasets\data*

Τα αρχεία κώδικα (***DriverPredict_(K-means).py, DriverPredictSenarios_(K-means).py, preprocessData.py, datasetEvaluation.py***), πρέπει να βρίσκονται τοπικά στην τοποθεσία *C:\datasets\code*

Θα πρέπει να υπάρχει ο φάκελος *C:\datasets\plots* καθώς και ο φάκελος *C:\datasets\eval\plots*, όπου θα αποθηκεύονται αυτόματα τα διαγράμματα που θα παράγονται από τον κώδικα.

preprocessData.py

```
from pathlib import Path
import pandas as pd
import os
import sys
import zipfile
from datetime import datetime
from tzlocal import get_localzone
import numpy as np

MORNING_START   = datetime.strptime('05::00::00', '%H::%M::%S').time()
MORNING_END     = datetime.strptime('11::59::59', '%H::%M::%S').time()
AFTERNOON_START = datetime.strptime('12::00::00', '%H::%M::%S').time()
AFTERNOON_END   = datetime.strptime('16::59::59', '%H::%M::%S').time()
EVENING_START   = datetime.strptime('17::00::00', '%H::%M::%S').time()
EVENING_END     = datetime.strptime('20::59::59', '%H::%M::%S').time()
NIGHT_START     = datetime.strptime('21::00::00', '%H::%M::%S').time()
NIGHT_END       = datetime.strptime('23::59::59', '%H::%M::%S').time()

def check_time(timeStart, timeEnd):
    if timeStart >= MORNING_START and timeEnd <= MORNING_END:
        return 0.25
    elif timeStart >= AFTERNOON_START and timeEnd <= AFTERNOON_END:
        return 0.50
    elif timeStart >= EVENING_START and timeEnd <= EVENING_END:
        return 0.75
    elif timeStart >= NIGHT_START and timeEnd <= NIGHT_END:
        return 1.00
    else:
        return '?'

def extract_zip(zip_filename, extract_path):
      try:
        with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
          zip_ref.extractall(extract_path)
        print("Extraction completed successfully.")
      except zipfile.BadZipFile:
        print("Error: File is not a valid ZIP file.")
      except FileNotFoundError as e:
        print(f"Error: {e}")
        print(f"File not found: {zip_filename}")
      except Exception as e:
        print(f"An unexpected error occurred: {e}")

def load_data(sensor, file):
    if sensor=="Accelerometer":
        colnames = ['Date', 'Ms', 'AccX', 'AccY', 'AccZ']
        datatypes = {'Ms': 'int32', 'AccX': 'float32', 'AccY': 'float32','AccZ': 'float32'}
    elif sensor=="Gyroscope":
        colnames = ['Date', 'Ms', 'GyroX', 'GyroY', 'GyroZ']
        datatypes = {'Ms': 'int32', 'GyroX': 'float32', 'GyroY': 'float32','GyroZ': 'float32'}

    dfTemp = pd.read_csv(file, header=0, names=colnames, dtype=datatypes,
                         parse_dates=['Date'], dayfirst=True)

    timeStart = dfTemp['Date'].iloc[0].time()
    timeEnd = dfTemp['Date'].iloc[len(dfTemp.index) - 1].time()
    driveTime = check_time(timeStart, timeEnd)
    dfTemp.insert(loc=0, column='DayPart', value=driveTime)

    pos = str(file).find("Day-")
    driver = int(str(file)[ pos + 4 : pos + 5])
    dfTemp.insert(loc=0, column='Driver', value=driver)

    # Get the local timezone
    local_timezone = get_localzone()

    dfTemp['Date'] = pd.to_datetime(dfTemp['Date']).dt.tz_localize(local_timezone)
    df1=(dfTemp['Date'].astype(np.int64) // 10**6) + dfTemp['Ms']
    dfTemp.insert(loc=4, column='Millsecs', value=df1)


    if sensor=="Accelerometer":
        dfTemp['GyroX']=np.nan
        dfTemp['GyroY']=np.nan
        dfTemp['GyroZ']=np.nan
    elif sensor=="Gyroscope":
        dfTemp.insert(loc=4, column='AccZ', value=np.nan)
        dfTemp.insert(loc=4, column='AccY', value=np.nan)
        dfTemp.insert(loc=4, column='AccX', value=np.nan)
    return dfTemp
```

```
if not os.path.isdir('C:\datasets\data'):
      zip_filename = "C:\datasets\data.zip"
      extract_path = "C:\datasets"
      extract_zip(zip_filename, extract_path)

sensors=['Accelerometer', 'Gyroscope']
df_list = list()
i=1
folders_list = os.listdir(Path("c:/datasets/data"))

for x in folders_list:
    dfItem_list = list()
    print("\nLoad Sensors data from c:/datasets/data/"+x+" ("+str(i)+" of "+
          str(len(folders_list))+")...\n")

    for sensor in sensors:
        file=Path("c:/datasets/data/"+x+"/"+sensor+".csv")
        if file.exists():
            print("FIle "+x+"/"+sensor+".csv...")
            dfTemp=load_data(sensor, file)
            dfItem_list.append(dfTemp)
            del dfTemp

    numFiles=len(dfItem_list)
    if numFiles==2:
        dfItem_list[0] = dfItem_list[0].sort_values(by=["Driver", "DayPart", "Date", "Ms"],
                                                    ignore_index=True)
        dfItem_list[1] = dfItem_list[1].sort_values(by=["Driver", "DayPart", "Date", "Ms"],
                                                    ignore_index=True)

        print("Accelerometer rows: " + str(len(dfItem_list[0].index)))
        print("Gyroscope rows....: " + str(len(dfItem_list[1].index)))

        print("Merging Accelerometer and Gyroscope Sensors data...\n")
        df = pd.concat([dfItem_list[0], dfItem_list[1]])

        df = df.sort_values(by=["Driver", "DayPart", "Date","Ms"], ignore_index=True)

        dfGroup = df.groupby(["Driver", "DayPart", 'Date', 'Ms'], as_index=False)
        df=dfGroup.max()

        cols = ['Driver', 'DayPart', 'AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']
        df.loc[:,cols] = df.loc[:,cols].ffill()
        df.loc[:,cols] = df.loc[:,cols].bfill()

        df.drop_duplicates(keep='first', inplace=True, ignore_index=True)

        print("Merged rows....: " + str(len(df.index)))

        print("\nFolder "+x+":\n")
        print("Number of rows: "+str(len(df.index)) + "\n")

        df_list.append(df)
        del df
    dfItem_list.clear()
    i=i+1


print("\nConcatenate All Dataframes...\n")
df = pd.concat(df_list).reset_index(drop=True)
df = df.sort_values(by=["Driver", "DayPart", "Date","Ms"], ignore_index=True)

print(df.info())
print("Concat rows....: " + str(len(df.index)))

print("Save data locally as finelData.csv...\n")
df.to_csv('C:/datasets/finalData.csv.zip', chunksize=20000, index=False,
          compression='zip')

print("Finish!")
sys.exit()
```

datasetEvaluation.py

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import zipfile
import scipy.stats as stats
from scipy.stats import norm
import seaborn as sns
import dataframe_image as dfi
import sys

filepath='C:/datasets/'
filename='finalData.csv'
plotname='Sensors'

print(filepath, filename, plotname)
#---  Read Data from Zip CSV file  ------------------------------
zf = zipfile.ZipFile(filepath+filename+'.zip')
df = pd.read_csv(zf.open(filename))
#----------------------------------------------------------------

# Display the dimensions of the dataset
print("\nShape of dataset:\n", df.shape)

# Display the first few rows of the dataset
print("\nFirst 20 rows of the dataset:\n", df.head(20))

# Check for missing values
print("\nMissing values in the dataset:\n", df.isnull().sum())

# Check data types of columns
print("\nData types of columns:\n", df.dtypes)

# Check for duplicate rows
duplicate_rows = df[df.duplicated()]
print("\nDuplicate rows in the dataset:\n", duplicate_rows)


# Statistical summary
dfSensorsData = df[['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']]

print('Summary statistics:\n', dfSensorsData.describe())
dfi.export(dfSensorsData.describe(), 'C:/datasets/plots/DatasetDescribe.png')

# Calculate skewness of each feature
skewness = dfSensorsData.skew()
print("\nSkewness of each feature (Skewness=0 for normal distribution):\n", skewness)

# Calculate kurtosis of each feature
kurtosis = dfSensorsData.kurtosis()
print("Kurtosis of each feature (kurtosis=3 for normal distribution):\n", kurtosis)
print('\n')

#---  Selection of Sensors Features  ------------------
answer = input("\nChoose Accelerator Features (XYZ): ")
answer=answer.upper()
fAcc = []
if all(char in 'XYZ' for char in answer):
    for axis in answer:
        fAcc.append('Acc'+axis)
else:
    print('Wrong Answer')
    sys.exit()
print("you chose: " + str(fAcc) + "\n")

answer = input("\nChoose Gyroscope Features (XYZ): ")
answer=answer.upper()
fGyro = []
if all(char in 'XYZ' for char in answer):
    for axis in answer:
        fGyro.append('Gyro'+axis)
else:
    print('Wrong Answer')
    sys.exit()
print("you chose: " + str(fGyro) + "\n")
#-------------------------------------------------------------------------------------------

dfAcc = dfSensorsData[fAcc]
dfGyro = dfSensorsData[fGyro]

plt.scatter(dfAcc['AccY'], dfGyro['GyroZ'])
```

```
plt.show()

# Find out if data follows a Gaussian distribution
#===  Plot histograms  ========================================
# Accelerator
plt.figure(figsize=(15, 8))
for i, axis in enumerate(fAcc):
    plt.subplot(2, 3, i + 1)
    num_bins = int(np.sqrt(len(dfAcc[axis])))  # Square root choice
    sns.histplot(dfAcc[axis], stat='frequency', bins=num_bins, color='blue')
    plt.title(f'Accelerometer {axis}')
    plt.xlabel('Value')
    plt.ylabel('Frequency')

# Gyroscope
for i, axis in enumerate(fGyro):
    plt.subplot(2, 3, i + 4)
    num_bins = int(np.sqrt(len(dfGyro[axis])))  # Square root choice
    sns.histplot(dfGyro[axis], stat='frequency', bins=num_bins, color='green')
    plt.title(f'Gyroscope {axis}')
    plt.xlabel('Value')
    plt.ylabel('Frequency')

pltTitle = 'Evaluation Histogram Plots (' + plotname + ')'
plt.suptitle(pltTitle)
plt.subplots_adjust(top = 0.9, bottom = 0.1, wspace=0.3, hspace=0.3)
plt.savefig(filepath+"eval/plots/" + pltTitle + ".png")
plt.show()

#=================================================================

#==  Plot Q-Q plots  ======================================
plt.figure(figsize=(12, 6))
for i, axis in enumerate(fAcc):
    plt.subplot(2, 3, i + 1)
    stats.probplot(dfAcc[axis], dist="norm", plot=plt)
    plt.title(f'Accelerometer Q-Q Plot ({axis})')

# Plot Q-Q plots for gyroscope data
for i, axis in enumerate(fGyro):
    plt.subplot(2, 3, i + 4)
    stats.probplot(dfGyro[axis], dist="norm", plot=plt)
    plt.title(f'Gyroscope Q-Q Plot ({axis})')
pltTitle = 'Evaluation Q-Q Plots (' + plotname + ')'
plt.suptitle(pltTitle)
plt.subplots_adjust(top = 0.9, wspace=0.3, hspace=0.5)
plt.savefig(filepath+"eval/plots/" + pltTitle + ".png")
plt.show()
#=================================================================

# Find out if data contains Outliers
#==  Create a box plot  ======================================
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.boxplot(data=dfAcc)
plt.title('Accelerometer Box Plot')
plt.ylabel('Value')

# Create a box plot for gyroscope data
plt.subplot(1, 2, 2)
sns.boxplot(data=dfGyro)
plt.title('Gyroscope Box Plot')
plt.ylabel('Value')
pltTitle = 'Evaluation Box Plots (' + plotname + ')'
plt.suptitle(pltTitle)
plt.tight_layout()
plt.savefig(filepath+"eval/plots/" + pltTitle + ".png")
plt.show()

# Identify Acc outliers using Q5 and Q95
q5 = dfAcc.quantile(0.05)
q95 = dfAcc.quantile(0.95)
iqr = q95 - q5
lower_bound = q5 - 1.5 * iqr
upper_bound = q95 + 1.5 * iqr
outliers = ((dfAcc < lower_bound) | (dfAcc > upper_bound)).any(axis=1)
print("Acc Outliers:")
print(dfAcc[outliers])
dfAcc[outliers].to_csv('C:/datasets/eval/outliersAcc.csv', index=False)
# Identify Gyro outliers using Q5 and Q95
q5 = dfGyro.quantile(0.05)
q95 = dfGyro.quantile(0.95)
iqr = q95 - q5
lower_bound = q5 - 1.5 * iqr
upper_bound = q95 + 1.5 * iqr
outliers = ((dfGyro < lower_bound) | (dfGyro > upper_bound)).any(axis=1)
```

```
print("Gyro Outliers:")
print(dfGyro[outliers])
dfGyro[outliers].to_csv('C:/datasets/eval/outliersGyro.csv', index=False)
#================================================================
```

DriverPredictSenarios_(K-means.py)

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import normalize
from scipy.stats.mstats import winsorize
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
import zipfile
import sys
import os
from pathlib import Path


def outliersDetection(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1:
        # Apply Winsorizer to data
        df1 = df
        for column in features:
            df1.loc[:, column] = winsorize(df1[column], limits=[0.01, 0.01])
        plotTitle.append('Winsorizer')
    return df1

def normalization(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1 or answer==2 or answer==3:
        if answer==1: norm='l1'
        if answer==2: norm='l2'
        if answer==3: norm='max'
        # Apply Normalization to data
        df1 = normalize(df, norm=norm)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Normalization ' + norm)
    return df1

def featuresScaling(features, preProp, df):
    if preProp==0:
        df1 = df
    elif preProp==1:
        # Standardize the features for clustering
        scaler = StandardScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Standardize')
    elif preProp==2:
        # Apply MinMaxScaler to data
        scaler = MinMaxScaler(feature_range=(-1, 1))
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('MinMaxScaler')
    elif preProp==3:
        # Apply RobustScaler to data
        scaler = RobustScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('RobustScaler')
    return df1

def evaluation():
    inertia = []
    silhouette_scores = []
    calinski_harabasz_scores = []
    davies_bouldin_scores = []
    labels_per_cluster = {}
    cluster_centers_per_cluster = {}
    inertia_per_cluster = {}
    silhouette_scores_per_cluster = {}
    calinski_harabasz_scores_per_cluster = {}
    davies_bouldin_scores_per_cluster = {}

    cluster_range = range(2, 11)
```

```
    for k in cluster_range:
        kmeans = KMeans(n_clusters=k, init="k-means++", random_state=45)
        cluster_labels = kmeans.fit_predict(dfFinal)

        labels_per_cluster[k] = cluster_labels
        inertia_per_cluster[k] = kmeans.inertia_
        cluster_centers_per_cluster[k] = kmeans.cluster_centers_
        inertia.append(inertia_per_cluster[k])
        silhouette_scores_per_cluster[k] = silhouette_score(dfFinal,
                                                            cluster_labels,
                                                            sample_size=score_sample,
                                                            random_state=45)
        silhouette_scores.append(silhouette_scores_per_cluster[k])
        calinski_harabasz_scores_per_cluster[k] = calinski_harabasz_score(dfFinal,
                                                                          cluster_labels)
        calinski_harabasz_scores.append(calinski_harabasz_scores_per_cluster[k])
        davies_bouldin_scores_per_cluster[k] = davies_bouldin_score(dfFinal, cluster_labels)
        davies_bouldin_scores.append(davies_bouldin_scores_per_cluster[k])



    # Plot evaluation metrics
    fig, ax = plt.subplots(2, 2, figsize=(12, 10))

    # Inertia
    ax[0, 0].plot(cluster_range, inertia, marker='o', color='red')
    ax[0, 0].set_title('Elbow Method', color='red')
    ax[0, 0].set_xlabel('Number of Clusters')
    ax[0, 0].set_ylabel('Inertia')

    # Silhouette Score
    ax[0, 1].plot(cluster_range, silhouette_scores, marker='o', color='green')
    ax[0, 1].set_title('Silhouette Score', color='green')
    ax[0, 1].set_xlabel('Number of Clusters')
    ax[0, 1].set_ylabel('Silhouette Score')

    # Davies-Bouldin Index
    ax[1, 0].plot(cluster_range, davies_bouldin_scores, marker='o', color='purple')
    ax[1, 0].set_title('Davies-Bouldin Index', color='purple')
    ax[1, 0].set_xlabel('Number of Clusters')
    ax[1, 0].set_ylabel('Davies-Bouldin Index')

    # Calinski-Harabasz Index
    ax[1, 1].plot(cluster_range, calinski_harabasz_scores, marker='o', color='blue')
    ax[1, 1].set_title('Calinski-Harabasz Index', color='blue')
    ax[1, 1].set_xlabel('Number of Clusters')
    ax[1, 1].set_ylabel('Calinski-Harabasz Index')


    pltTitle = createPlotTitle(plotTitle)
    pltTitle = 'Evaluation (' + pltTitle + ')'
    plt.suptitle(pltTitle)
    plt.savefig("c:/datasets/plots/evalNumClusters/" + pltTitle + ".png")
    plt.subplots_adjust(top = 0.9, bottom = 0.1, hspace=0.5)
    plt.show()

    #---  Based on the evaluation, set the optimal number of clusters  ------------------
    k_optimal = int(input("Senario " + str(ind) + ": Enter the number of optimal Clusters:"))
    #-------------------------------------------------------------------------------------

    inertiaVal      = round(inertia_per_cluster[k_optimal],3)
    silhouetteVal   = round(silhouette_scores_per_cluster[k_optimal],3)
    calinskiVal     = round(calinski_harabasz_scores_per_cluster[k_optimal],3)
    daviesVal       = round(davies_bouldin_scores_per_cluster[k_optimal],3)

    pltTitle = createPlotTitle(plotTitle)
    if senarioAnswer==1:
        sign='s'
    elif senarioAnswer==2:
        sign='r'

    record = [sign+senario,
              pltTitle,
              k_optimal,
              inertiaVal,
              silhouetteVal,
              calinskiVal,
              daviesVal
              ]
    dfMetrics.loc[len(dfMetrics)] = record

    # Print the record with the best total score
    print("Senario " + str(ind) + ": Best record:",
          (k_optimal,
           inertiaVal,
           silhouetteVal,
```

```
            calinskiVal,
            daviesVal
            ))
    print('\n')

    return k_optimal, labels_per_cluster[k_optimal], cluster_centers_per_cluster[k_optimal]


def methodPCA(df_scaled, num):
    # Create PCA object for Axis X
    pca_X = PCA(n_components=num)
    pcaAcc = pca_X.fit_transform(df_scaled[['AccX', 'AccY', 'AccZ']])

    # Create PCA object for Axis Y
    pca_Y = PCA(n_components=num)
    pcaGyro = pca_Y.fit_transform(df_scaled[['GyroX', 'GyroY', 'GyroZ']])

    # Concatenate PCA results
    dfFinal = pd.DataFrame({
        'Acc': pcaAcc.flatten(),
        'Gyro': pcaGyro.flatten()
    })
    plotTitle.append('PCA')

    return dfFinal

def filterData(df):
    flt_s = input("\nFilter application (0/1/2/3)\n"+
                            "0: None\n"+
                            "1: Driver\n"+
                            "2: DayPart\n"+
                            "3: Driver & DayPart\n"+
                            "Enter choice: ")
    print("you chose: " + flt_s + "\n")
    flt = int(flt_s)

    if flt==0:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle1.append('All')
    elif flt==1:
        print("\nDataset will be filtered for some driver!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        dfFiltered = df.loc[df['Driver'] == flt1]
        plotTitle1.append('Driver'+flt1_s)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['DayPart']))
            print('\nDriver'+flt1_s+' made ' + str(n) + ' roots\n')
    elif flt==2:
        print("\nDataset will be filtered for some part of the Day!\n")
        flt1_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt1_s + "\n")
        flt1 = float(flt1_s)
        dfFiltered = df.loc[df['DayPart'] == flt1]
        if flt1==0.25:
            title1="Morning"
        elif flt1==0.5:
            title1="Afternoon"
        elif flt1==0.75:
            title1="Evening"
        elif flt1==1.0:
            title1="Night"
        else:
            title1="All"
        plotTitle1.append('Part of Day '+title1)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['Driver']))
            print('\nThe Part of Day '+flt1_s+' made by ' + str(n) + ' Drivers\n')
    elif flt==3:
        print("\nDataset will be filtered for a compination of Driver AND part of the Day!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        title1 = 'Driver'+flt1_s
        flt2_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
```

```
                    "Enter choise: ")
        print("you chose: " + flt2_s + "\n")
        flt2 = float(flt2_s)
        dfFiltered = df[(df.Driver == flt1) & (df.DayPart == flt2)]
        if flt2==0.25:
            title2="Morning"
        elif flt2==0.5:
            title2="Afternoon"
        elif flt2==0.75:
            title2="Evening"
        elif flt2==1.0:
            title2="Night"
        else:
            title2="All"
        plotTitle1.append(title1 + ' and ' + title2)
    else:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle1.append('All')

    row_count = len(dfFiltered)

    return dfFiltered, row_count

def createPlotTitle(plotTitle):
    pltTitle = ''
    for item in plotTitle:
        pltTitle = pltTitle + item + '-'
    pltTitle = pltTitle + 'K-means'
    return pltTitle


#---  Read Data from Zip CSV file  ------------------------------------------------------
file=Path("c:/datasets/finalData.csv.zip")
if not os.path.isfile(file):
    print("Dataset: C:/Datasets/finalData.csv.zip not exists!")
    sys.exit()
zf = zipfile.ZipFile('C:/Datasets/finalData.csv.zip')
df = pd.read_csv(zf.open('finalData.csv'))
#-----------------------------------------------------------------------------------------

#---  Read Senarios Data from CSV file  -------------------------------------------------
senarioAnswer = int(input("Choose Senarios Group (1/2):\n"+
                          "1: Normalization-Features Scaling\n"+
                          "2: Features Scaling-Normalization\n"+
                          "Enter Choise: "))
if senarioAnswer==1:
    file=Path("c:/datasets/executionSenarios.csv")
    if not os.path.isfile(file):
        print("Dataset: C:/Datasets/executionSenarios.csv not exists!")
        sys.exit()
    dfSenarios = pd.read_csv('C:\datasets\executionSenarios.csv')
elif senarioAnswer==2:
    file=Path("c:/datasets/executionSenarios2.csv")
    if not os.path.isfile(file):
        print("Dataset: C:/Datasets/executionSenarios2.csv not exists!")
        sys.exit()
    dfSenarios = pd.read_csv('C:\datasets\executionSenarios2.csv')
else:
    print("Wrong Choise!")
    sys.exit()
#-----------------------------------------------------------------------------------------

features = ['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']

plotTitle1 = []

#---  Filtering  -----------------------------------------------------------------------
dfFiltered, row_count = filterData(df)
print(f"The Filtered Data have {row_count} Rows!")
if row_count <= 0:
    sys.exit()
df_scaled = dfFiltered[features]
# Find the more appropriate number of samples for silhouette score
score_sample = round(row_count * 5 / 100)
#-----------------------------------------------------------------------------------------

dfMetrics = pd.DataFrame(columns=['senario',
                                  'title',
                                  'clusters',
                                  'inertia',
                                  'silhouette',
                                  'calinskiHarabasz',
                                  'davisBouldin'])

for ind in dfSenarios.index:
```

```
    plotTitle = []
    plotTitle.append(plotTitle1[0])

    winsorizeVal        = dfSenarios['Winsorize'][ind]
    l1Val               = dfSenarios['L1'][ind]
    l2Val               = dfSenarios['L2'][ind]
    maxVal              = dfSenarios['Max'][ind]
    standardScalerVal   = dfSenarios['StandardScaler'][ind]
    minMaxScalerVal     = dfSenarios['MinMaxScaler'][ind]
    robustScalerVal     = dfSenarios['RobustScaler'][ind]

    if senarioAnswer==1:
        senario = str(winsorizeVal) + str(l1Val) + str(l2Val) + str(maxVal)
        senario = senario + str(standardScalerVal) + str(minMaxScalerVal) + str(robustScalerVal)
    elif senarioAnswer==2:
        senario = str(winsorizeVal) + str(standardScalerVal) + str(minMaxScalerVal)
        senario = senario + str(robustScalerVal) + str(l1Val) + str(l2Val) + str(maxVal)
    processCount = senario.count('1')

    df_scaled = outliersDetection(features, winsorizeVal, df_scaled)
    if winsorizeVal==1:
        print("Senario " + str(ind) + ": You performed Outliers Detection with Winsorize!")
    #-------------------------------------------------------------------------------------

    if senarioAnswer==1:
        #---  Normalization Methods  ---------------------------------------------------
        if l1Val==1:
            answer=1
            print("Senario " + str(ind) + ": You performed Normalization with Normalize (L1)!")
        elif l2Val==1:
            answer=2
            print("Senario " + str(ind) + ": You performed Normalization with Normalize (L2)!")
        elif maxVal==1:
            answer=3
            print("Senario " + str(ind) + ": You performed Normalization with Normalize (Max)!")
        else:
            answer=0
        df_scaled = normalization(features, answer, df_scaled)
        #-------------------------------------------------------------------------------------

        #---  Features Scaling Methods  ------------------------------------------------
        if standardScalerVal==1:
            answer=1
            print("Senario " + str(ind) + ": You performed Features Scaling (StandardScaler)!")
        elif minMaxScalerVal==1:
            answer=2
            print("Senario " + str(ind) + ": You performed Features Scaling (MinMaxScaler)!")
        elif robustScalerVal==1:
            answer=3
            print("Senario " + str(ind) + ": You performed Features Scaling (RobustScaler)!")
        else:
            answer=0
        df_scaled = featuresScaling(features, answer, df_scaled)
        #-------------------------------------------------------------------------------------
    elif senarioAnswer==2:
        #---  Features Scaling Methods  ------------------------------------------------
        if standardScalerVal==1:
            answer=1
            print("Senario " + str(ind) + ": You performed Features Scaling (StandardScaler)!")
        elif minMaxScalerVal==1:
            answer=2
            print("Senario " + str(ind) + ": You performed Features Scaling (MinMaxScaler)!")
        elif robustScalerVal==1:
            answer=3
            print("Senario " + str(ind) + ": You performed Features Scaling (RobustScaler)!")
        else:
            answer=0
        df_scaled = featuresScaling(features, answer, df_scaled)
        #-------------------------------------------------------------------------------------

        #---  Normalization Methods  ---------------------------------------------------
        if l1Val==1:
            answer=1
            print("Senario " + str(ind) + ": You performed Normalization with Normalize (L1)!")
        elif l2Val==1:
            answer=2
            print("Senario " + str(ind) + ": You performed Normalization with Normalize (L2)!")
        elif maxVal==1:
            answer=3
            print("Senario " + str(ind) + ": You performed Normalization with Normalize (Max)!")
        else:
            answer=0
        df_scaled = normalization(features, answer, df_scaled)
        #-------------------------------------------------------------------------------------
```

```
    #---  PCA Method  ------------------------------------------------------------------------
    dfFinal = methodPCA(df_scaled, 1)
    #-------------------------------------------------------------------------------------------

    #---  Determine the optimal number of clusters using evaluation  -------------------------
    k_optimal, cluster_labels, cluster_centers = evaluation()
    #-------------------------------------------------------------------------------------------

if senarioAnswer==2:
    dfMetrics1 = pd.read_csv('C:\datasets\clustersMetrics.csv')
    dfMetrics = pd.concat([dfMetrics1, dfMetrics], ignore_index=True)

# Normalize scores values in scale
minVal = dfMetrics['inertia'].min()
maxVal = dfMetrics['inertia'].max()
inertiaNorm = (dfMetrics['inertia'] - minVal) / (maxVal - minVal)
minVal = dfMetrics['silhouette'].min()
maxVal = dfMetrics['silhouette'].max()
silhouetteNorm = (dfMetrics['silhouette'] - minVal) / (maxVal - minVal)
minVal = dfMetrics['calinskiHarabasz'].min()
maxVal = dfMetrics['calinskiHarabasz'].max()
calinskiHarabaszNorm = (dfMetrics['calinskiHarabasz'] - minVal) / (maxVal - minVal)
minVal = dfMetrics['davisBouldin'].min()
maxVal = dfMetrics['davisBouldin'].max()
davisBouldinNorm = (dfMetrics['davisBouldin'] - minVal) / (maxVal - minVal)


# Calculate total score for each cluster number
dfMetrics['totalScore'] = silhouetteNorm + calinskiHarabaszNorm - inertiaNorm - davisBouldinNorm

#==  Create Metrics Plot  ================================================
# Get unique clusters
clusters = dfMetrics['clusters'].unique()
nums = len(clusters)
# Generate colors for clusters
colors = plt.cm.tab10(np.linspace(0, 1, nums))

plt.figure(figsize=(14, 7))
# Plot bars for each cluster
for i, cluster in enumerate(clusters):
    clusterNums = dfMetrics[dfMetrics['clusters'] == cluster]
    plt.bar(clusterNums['senario'], clusterNums['totalScore'], color=colors[i],
            label=f'Clusters {cluster}')
plt.title('Metrics Scores (Inertia, Silhouette, Calinski Harabasz, Davis Bouldin) Plot')
plt.xlabel('Senario')
plt.ylabel('Total Score')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='dotted')
plt.legend()
plt.tight_layout()
plt.savefig("c:/datasets/plots/evalNumClusters/clustersMetrics.png")
plt.show()
#=========================================================================

print("Save Metrics Data locally as clustersMetrics.csv...\n")
dfMetrics.to_csv('C:/datasets/clustersMetrics.csv', index=False)
```

DriverPredict_(K-means).py

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import normalize
from scipy.stats.mstats import winsorize
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
import zipfile
import seaborn as sns
import sys
import dataframe_image as dfi
from PIL import Image, ImageDraw, ImageFont

plotTitle = []

def outliersDetection(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1:
        # Apply Winsorizer to data
        df1 = df
        for column in features:
            df1.loc[:, column] = winsorize(df1[column],
                                           limits=[0.01, 0.01])
        plotTitle.append('Winsorizer')
    return df1

def normalization(features, answer, df):
    if answer==0:
        df1 = df
    elif answer==1 or answer==2 or answer==3:
        if answer==1: norm='l1'
        if answer==2: norm='l2'
        if answer==3: norm='max'
        # Apply Normalization to data
        df1 = normalize(df, norm=norm)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Normalization ' + norm)
    return df1

def featuresScaling(features, preProp, df):
    if preProp==0:
        df1 = df
    elif preProp==1:
        # Standardize the features for clustering
        scaler = StandardScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('Standardize')
    elif preProp==2:
        # Apply MinMaxScaler to data
        scaler = MinMaxScaler(feature_range=(-1, 1))
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('MinMaxScaler')
    elif preProp==3:
        # Apply RobustScaler to data
        scaler = RobustScaler()
        df1 = scaler.fit_transform(df)
        df1=pd.DataFrame(df1, columns=features)
        plotTitle.append('RobustScaler')
    return df1

def evaluation():
    inertia = []
    silhouette_scores = []
    calinski_harabasz_scores = []
    davies_bouldin_scores = []
    labels_per_cluster = {}
    cluster_centers_per_cluster = {}
    inertia_per_cluster = {}
    silhouette_scores_per_cluster = {}
    calinski_harabasz_scores_per_cluster = {}
    davies_bouldin_scores_per_cluster = {}
```

```
cluster_range = range(2, 11)

for k in cluster_range:
    kmeans = KMeans(n_clusters=k, init="k-means++", random_state=45)
    cluster_labels = kmeans.fit_predict(dfFinal)

    labels_per_cluster[k] = cluster_labels
    inertia_per_cluster[k] = kmeans.inertia_
    cluster_centers_per_cluster[k] = kmeans.cluster_centers_
    inertia.append(inertia_per_cluster[k])
    silhouette_scores_per_cluster[k] = silhouette_score(dfFinal,
                                                        cluster_labels,
                                                        sample_size=score_sample,
                                                        random_state=45)
    silhouette_scores.append(silhouette_scores_per_cluster[k])
    calinski_harabasz_scores_per_cluster[k] = calinski_harabasz_score(dfFinal,
                                                                      cluster_labels)
    calinski_harabasz_scores.append(calinski_harabasz_scores_per_cluster[k])
    davies_bouldin_scores_per_cluster[k] = davies_bouldin_score(dfFinal, cluster_labels)
    davies_bouldin_scores.append(davies_bouldin_scores_per_cluster[k])


# Plot evaluation metrics
fig, ax = plt.subplots(2, 2, figsize=(12, 10))

# Inertia
ax[0, 0].plot(cluster_range, inertia, marker='o', color='red')
ax[0, 0].set_title('Elbow Method', color='red')
ax[0, 0].set_xlabel('Number of Clusters')
ax[0, 0].set_ylabel('Inertia')

# Silhouette Score
ax[0, 1].plot(cluster_range, silhouette_scores, marker='o', color='green')
ax[0, 1].set_title('Silhouette Score', color='green')
ax[0, 1].set_xlabel('Number of Clusters')
ax[0, 1].set_ylabel('Silhouette Score')

# Davies-Bouldin Index
ax[1, 0].plot(cluster_range, davies_bouldin_scores, marker='o', color='purple')
ax[1, 0].set_title('Davies-Bouldin Index', color='purple')
ax[1, 0].set_xlabel('Number of Clusters')
ax[1, 0].set_ylabel('Davies-Bouldin Index')

# Calinski-Harabasz Index
ax[1, 1].plot(cluster_range, calinski_harabasz_scores, marker='o', color='blue')
ax[1, 1].set_title('Calinski-Harabasz Index', color='blue')
ax[1, 1].set_xlabel('Number of Clusters')
ax[1, 1].set_ylabel('Calinski-Harabasz Index')


pltTitle = createPlotTitle(plotTitle)
pltTitle = 'Evaluation (' + pltTitle + ')'
plt.suptitle(pltTitle)
plt.savefig("c:/datasets/plots/" + pltTitle + ".png")
plt.subplots_adjust(top = 0.9, bottom = 0.1, hspace=0.5)
#plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=1.0)
plt.show()

#---  Based on the evaluation, choose the optimal number of clusters  ------------------
k_str = input("Enter the number of optimal Clusters:")
print("Number of Clusters is: " + k_str + "\n")
k_optimal = int(k_str)
#-------------------------------------------------------------------------------------------
dfClustersCount = pd.DataFrame(pd.DataFrame(labels_per_cluster[k_optimal]).value_counts())
dfClustersCount = dfClustersCount.rename(columns={'0': 'Cluster', 'count': 'Records'})
print('Recs per Cluster:\n', dfClustersCount)
dfi.export(dfClustersCount, 'C:/datasets/plots/ClustersCount.png')

Inertia                = round(inertia_per_cluster[k_optimal],3)
SilhouetteScore        = round(silhouette_scores_per_cluster[k_optimal],3)
CalinskiHarabaszScore  = round(calinski_harabasz_scores_per_cluster[k_optimal],3)
DaviesBouldinScore     = round(davies_bouldin_scores_per_cluster[k_optimal],3)
print('Inertia: ', Inertia)
print('SilhouetteScore: ', SilhouetteScore)
print('CalinskiHarabaszScore: ', CalinskiHarabaszScore)
print('DaviesBouldinScore: ', DaviesBouldinScore)
#---  Get the metrics
metrics = pd.DataFrame({
    'Inertia': [Inertia],  # Scalar values wrapped in a list
    'SilhouetteScore': [SilhouetteScore],
    'CalinskiHarabaszScore': [CalinskiHarabaszScore],
    'DaviesBouldinScore': [DaviesBouldinScore]
}, index=[k_optimal])
print('\nEvaluation Metrics:\n', metrics)
pltTitle = createPlotTitle(plotTitle)
```

```
    pltTitle = 'Metrics (' + str(k_optimal) + ') clusters (' + pltTitle + ')'
    dfi.export(metrics, 'C:/datasets/plots/' + pltTitle + '.png')

    return k_optimal, labels_per_cluster[k_optimal], cluster_centers_per_cluster[k_optimal]


def methodPCA(df_scaled, featuresX, featuresY, num):
    # Create PCA object for Axis X
    if len(featuresX)>1:
        pca_X = PCA(n_components=num)
        pcaX = pca_X.fit_transform(df_scaled[featuresX])
    else:
        pcaX = df_scaled[featuresX].values

    # Create PCA object for Axis Y
    if len(featuresY)>1:
        pca_Y = PCA(n_components=num)
        pcaY = pca_Y.fit_transform(df_scaled[featuresY])
    else:
        pcaY = df_scaled[featuresY].values

    # Concatenate PCA results
    dfFinal = pd.DataFrame({
        'axisX': pcaX.flatten(),
        'axisY': pcaY.flatten()
    })
    plotTitle.append('PCA')

    return dfFinal

def filterData(df):
    flt_s = input("\nFilter application (0/1/2/3)\n"+
                              "0: None\n"+
                              "1: Driver\n"+
                              "2: DayPart\n"+
                              "3: Driver & DayPart\n"+
                              "Enter choice: ")
    print("you chose: " + flt_s + "\n")
    flt = int(flt_s)

    if flt==0:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle.append('All')
    elif flt==1:
        print("\nDataset will be filtered for some driver!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        dfFiltered = df.loc[df['Driver'] == flt1]
        plotTitle.append('Driver'+flt1_s)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['DayPart']))
            print('\nDriver'+flt1_s+' made ' + str(n) + ' roots\n')
    elif flt==2:
        print("\nDataset will be filtered for some part of the Day!\n")
        flt1_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt1_s + "\n")
        flt1 = float(flt1_s)
        dfFiltered = df.loc[df['DayPart'] == flt1]
        if flt1==0.25:
            title1="Morning"
        elif flt1==0.5:
            title1="Afternoon"
        elif flt1==0.75:
            title1="Evening"
        elif flt1==1.0:
            title1="Night"
        else:
            title1="All"
        plotTitle.append('Part of Day '+title1)
        if len(dfFiltered)>0:
            n = len(pd.unique(dfFiltered['Driver']))
            print('\nThe Part of Day '+flt1_s+' made by ' + str(n) + ' Drivers\n')
    elif flt==3:
        print("\nDataset will be filtered for a compination of Driver AND part of the Day!\n")
        flt1_s = input("\nEnter Driver number (1-7): ")
        print("you chose: " + flt1_s + "\n")
        flt1 = int(flt1_s)
        title1 = 'Driver'+flt1_s
        flt2_s = input("\nEnter part of the Day as number (0.25/0.5/0.75/1.0):\n" +
```

```
                    "0.25 - Morning   (07:00:00 - 11:59:59)\n" +
                    "0.5  - Afternoon (12:00:00 - 16:59:59)\n" +
                    "0.75 - Evening   (17:00:00 - 20:59:59)\n" +
                    "1.0  - Night     (21:00:00 - 23:59:59)\n" +
                    "Enter choise: ")
        print("you chose: " + flt2_s + "\n")
        flt2 = float(flt2_s)
        dfFiltered = df[(df.Driver == flt1) & (df.DayPart == flt2)]
        if flt2==0.25:
            title2="Morning"
        elif flt2==0.5:
            title2="Afternoon"
        elif flt2==0.75:
            title2="Evening"
        elif flt2==1.0:
            title2="Night"
        else:
            title2="All"
        plotTitle.append(title1 + ' and ' + title2)
    else:
        print("\nDataset will not be filtered (All Data)!\n")
        dfFiltered = df
        plotTitle.append('All')

    row_count = len(dfFiltered)

    return dfFiltered, row_count

def createPlotTitle(plotTitle):
    pltTitle = ''
    for item in plotTitle:
        pltTitle = pltTitle + item + '-'
    pltTitle = pltTitle + 'K-means Clustering'
    return pltTitle


#---  Read Data from Zip CSV file  ----------------------------------------------------
zf = zipfile.ZipFile('C:/Datasets/finalData.csv.zip')
df = pd.read_csv(zf.open('finalData.csv'))
#--------------------------------------------------------------------------------------

#---  Selection of Sensors Features  ------------------
answer = input("\nChoose Accelerator Features (XYZ): ")
answer=answer.upper()
fAcc = []
if all(char in 'XYZ' for char in answer):
    for axis in answer:
        fAcc.append('Acc'+axis)
print("you chose: " + str(fAcc) + "\n")

answer = input("\nChoose Gyroscope Features (XYZ): ")
answer=answer.upper()
fGyro = []
if all(char in 'XYZ' for char in answer):
    for axis in answer:
        fGyro.append('Gyro'+axis)
print("you chose: " + str(fGyro) + "\n")
featuresSensors=fAcc+fGyro
fTime=['Millsecs']
answer = input("\nHow will the two dimensions be separated? (1/2)\n"+
               "1: Accelerator - Gyroscope\n"+
               "2: Time - Sensors\n"+
               "Enter Choise: ")
features=[]
featuresX=[]
featuresY=[]
if all(char in '12' for char in answer):
    if answer=='2':
        features=fTime+featuresSensors
        featuresX=fTime
        featuresY=featuresSensors
        titleAxisX='Time'
        titleAxisY='Sensors ['
        for item in featuresSensors:
            titleAxisY=titleAxisY+item+', '
        titleAxisY=titleAxisY[:-2] + ']'
    else:
        features=featuresSensors
        featuresX=fAcc
        featuresY=fGyro
        titleAxisX='Accelerator ['
        titleAxisY='Gyroscope ['
        for item in featuresX:
            titleAxisX=titleAxisX+item+', '
        titleAxisX=titleAxisX[:-2] + ']'
        for item in featuresY:
```

```
            titleAxisY=titleAxisY+item+', '
        titleAxisY=titleAxisY[:-2] + ']'
else:
    print('Wrong Answer')
    sys.exit()
#-------------------------------------------------------------------------------------------

#---  Filtering  -----------------------------------------------------------------------
dfFiltered, row_count = filterData(df)
print(f"The Filtered Data have {row_count} Rows!")
if row_count <= 0:
    sys.exit()
df_scaled = dfFiltered[features]
# Find the more appropriate number of samples for silhouette score
score_sample = round(row_count * 5 / 100)
#-------------------------------------------------------------------------------------------

choise_str = input("\nWhat pre-processing method to perform? (0/1/2/3)\n"+
                           "0: None\n"+
                           "1: Outliers Detection\n"+
                           "2: Normalization\n"+
                           "3: Features Scaling\n"+
                           "Enter choice: ")
print("you chose: " + choise_str + "\n")
choise = int(choise_str)

while (choise!=0):

    #---  Outliers Detecting Methods  ------------------------------------------------------
    if choise!=1:
        answer = 0
    else:
        answer = 1
        print("You performed Outliers Detection with Winsorize!")

    df_scaled = outliersDetection(features, answer, df_scaled)
    #-------------------------------------------------------------------------------------------

    #---  Normalization Methods  -----------------------------------------------------------
    if choise!=2:
        answer = 0
    else:
        answer_str = input("\nPreferred Normalization technique (1/2/3)\n"+
                                   "1: Normalize (L1)\n"+
                                   "2: Normalize (L2)\n"+
                                   "3: Normalize (Max)\n"+
                                   "Enter choice: ")
        print("you chose: " + answer_str + "\n")
        answer = int(answer_str)
        print("You performed Normalization with Normaalize!")

    df_scaled = normalization(features, answer, df_scaled)
    #-------------------------------------------------------------------------------------------

    #---  Features Scaling Methods  --------------------------------------------------------
    if choise!=3:
        answer = 0
    else:
        answer_str = input("\nPreferred Features Scaling technique (0/1/2/3)\n"+
                                   "0: None\n"+
                                   "1: StandardScaler\n"+
                                   "2: MinMaxScaler\n"+
                                   "3: RobustScaler\n"+
                                   "Enter choice: ")
        print("you chose: " + answer_str + "\n")
        answer = int(answer_str)
        print("You performed Features Scaling!")

    df_scaled = featuresScaling(features, answer, df_scaled)
    #-------------------------------------------------------------------------------------------

    choise_str = input("\nWhat another pre-processing method to perform? (0/1/2/3)\n"+
                               "0: None\n"+
                               "1: Outliers Detection\n"+
                               "2: Normalization\n"+
                               "3: Features Scaling\n"+
                               "Enter choice: ")
    print("you chose: " + choise_str + "\n")
    choise = int(choise_str)

#---  PCA Method  ----------------------------------------------------------------------
dfFinal = methodPCA(df_scaled, featuresX, featuresY, 1)
#-------------------------------------------------------------------------------------------

#---  Determine the optimal number of clusters using evaluation  -------------------------
eval = input("Use evaluation to determine the optimal number of clusters? (y/n) :")
```

```
print("You entered: " + eval + "\n")
if eval=='y' or eval=='Y':
    k_optimal, cluster_labels, cluster_centers = evaluation()
else:
    k_optimal = int(input("Enter the optimal Clusters number: "))
    #---  Apply K-means clustering with the optimal number of clusters  -------------------
    kmeans = KMeans(n_clusters=k_optimal, init="k-means++", random_state=45)
    cluster_labels = kmeans.fit_predict(dfFinal)
    cluster_centers = kmeans.cluster_centers_
    #-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

#---  Add the cluster labels to the Final DataFrame
dfFinal['Cluster'] = cluster_labels

#---  Get the cluster centers
cluster_centers_combined = pd.DataFrame({
    'axisX': cluster_centers[:, 0],
    'axisY': cluster_centers[:, 1]
})
sorted_pairs = sorted(enumerate(zip(cluster_centers_combined['axisX'],
                                    cluster_centers_combined['axisY'])),
                                    key=lambda pair: abs(pair[1][0]) + abs(pair[1][1]))

output_string = "Clusters Centers Values:"
for index, (axisX, axisY) in sorted_pairs:
    output_string += f"\nCluster {index} --> axisX: {axisX:.3f} -- axisY: {axisY:.3f}"

#==  Set up image parameters  ===============================================================
font_size = 12
font_path = "arial.ttf"
font_color = (0, 0, 0)  # Black color for text
background_color = (255, 255, 255)  # White background color
padding = 10  # Padding around the text
# Create a font object
font = ImageFont.truetype(font_path, font_size)
# Create a drawing context
draw = ImageDraw.Draw(Image.new('RGB', (1, 1)))  # Create a temporary image
text_bbox = draw.textbbox((0, 0), output_string, font=font)
# Create a new image with white background
image_width = text_bbox[2] + 2 * padding
image_height = text_bbox[3] + 2 * padding
image = Image.new('RGB', (image_width, image_height), background_color)
draw = ImageDraw.Draw(image)
# Draw the wrapped text on the image
draw.text((padding, padding), output_string, font=font, fill=font_color)
# Save the image
pltTitle = createPlotTitle(plotTitle)
pltTitle = 'ClustersCenters (' + pltTitle + ')'
image.save('C:/datasets/plots/' + pltTitle + '.png')
#=============================================================================================

print(output_string)
print("\n")

#--- Name the Clusters for the Plot  ------------------------------------------------------
do_names = input("\nDo you want to name the Clusters? (y/n):")
if do_names=='y':
    sorted_pairs = sorted(enumerate(zip(cluster_centers_combined['axisX'],
                                        cluster_centers_combined['axisY'])),
                                        key=lambda pair: abs(pair[1][0]) + abs(pair[1][1]))
    labels = ['Normal', 'Aggresive', 'Dangerous']
    clabels=[sorted_pairs[0][0], sorted_pairs[1][0], sorted_pairs[2][0]]
    print(clabels)

    dfFinal['Cluster'] = np.where(dfFinal['Cluster']==clabels[0],labels[0],
                                  np.where(dfFinal['Cluster']==clabels[1],labels[1],

np.where(dfFinal['Cluster']==clabels[2],labels[2],'')))
#-------------------------------------------------------------------------------------------

#---  Define a color palette for the clusters
palette = sns.color_palette("hsv", k_optimal)

#---  Plot the data points after PCA and K-means clustering  -----------------------------
sns.set_style="whitegrid"
sns.scatterplot(data=dfFinal, x='axisX', y='axisY', hue='Cluster', palette=palette)
plt.scatter(cluster_centers_combined['axisX'], cluster_centers_combined['axisY'],
            c='red', marker='X', s=100, label='Centroids')
# Annotating the cluster centers with their coordinates
for       i,       (x,       y)       in       enumerate(zip(cluster_centers_combined['axisX'],
cluster_centers_combined['axisY'])):
    if i % 2 == 0:
        plt.text(x, y + 0.1, f'({x:.3f}, {y:.3f})',
                 fontsize=10, color='black', ha='left', va='bottom', weight='bold')
    else:
```

```
        plt.text(x, y + 0.1, f'({x:.3f}, {y:.3f})',
                 fontsize=10, color='black', ha='right', va='top', weight='bold')

pltTitle = createPlotTitle(plotTitle)
plt.title(pltTitle)
plt.xlabel(titleAxisX)
plt.ylabel(titleAxisY)

# Get legend handles and labels
handles, labels = plt.gca().get_legend_handles_labels()
# Format cluster labels in legend
formatted_labels = [f"{label} Cluster" if label != 'Centroids' else label for label in labels]
# Update legend with formatted labels
plt.legend(handles, formatted_labels, loc='upper right', fontsize='6')

plt.savefig("c:/datasets/plots/" + pltTitle + ".png")
plt.show()

#-------------------------------------------------------------------------------------------
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "Artificial Intelligence, A Modern Approach" (3rd edition) by Stuart J. Russell and Peter Norvig, 2010

[2] "Introduction to Artificial Intelligence" (2nd Edition) by Wolfgang Ertel, 2017

[3] "Introduction to Artificial Intelligence" by Mariusz Flasiński, Springer International Publishing Switzerland 2016

[4] "Machine Learning" by Tom M. Mitchell, 1997

[5] "Η επιστήμη των δεδομένων μέσα από τη γλώσσα R" by Βασίλειος Σ. Βερύκιος, Βασίλειος Καγκλής, Ηλίας Κ. Σταυρόπουλος, 2015

[6] "Επιχειρηματική Ευφυΐα και Εξόρυξη Δεδομένων" by Ευστάθιος Γ. Κύρκος, 2015

[7] "Introduction to Machine Learning" (2nd Edition) by Ethem Alpaydm, 2010

[8] "Introduction to Machine Learning" by Nils J. Nilsson, 1997

[9] "Artificial Neural Networks-An Introduction" by Kevin L. Priddy, Paul E. Keller, 2005

[10] "Deep Learning" by Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016

[11] Wawage, Pawan (2022), "Driver Behavior Detection Using Smartphone - Dataset", Mendeley Data, V2, doi: 10.17632/9vr83n7z5j.2

[12] Βιβλιοθήκη Python pathlip

[13] Βιβλιοθήκη Python os

[14] Βιβλιοθήκη Python zipfile

[15] Βιβλιοθήκη Python datetime

[16] Βιβλιοθήκη Python pandas

[17] Βιβλιοθήκη Python numpy

[18] Βιβλιοθήκη Python scikit-learn

[19] Βιβλιοθήκη Python scipy.stats.mstats.winsorize

[20] Βιβλιοθήκη Python seaborn

[21] Βιβλιοθήκη Python dataframe_image

[22] Βιβλιοθήκη Python matplotlib.pyplot

[23] "Exploratory Data Analysis" by John W. Tukey, 1977