ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Δεικτοδότησης Πολυδιάστατων Δεδομένων και Τεχνικές Μείωσης Δεδομένων για Αποτελεσματική Κατηγοριοποίηση Εγγύτερων Γειτόνων

Του φοιτητή
Δημήτριου Μανώλη
Αρ. Μητρώου: 174886

Επιβλέπων
Ουγιάρογλου Στέφανος
Εργαστηριακό Διδακτικό Προσωπικό

19 Ιουνίου 2021

Τίτλος Π.Ε: Μέθοδοι δεικτοδότησης πολυδιάστατων δεδομένων
και τεχνικές μείωσης δεδομένων για αποτελεσματική
κατηγοριοποίηση εγγύτερων γειτόνων
Κωδικός Π.Ε: 21164
Ονοματεπώνυμο φοιτητή: Δημήτριος Μανώλης
Ονοματεπώνυμο εισηγητή: Στέφανος Ουγιάρογλου
Ημερομηνία ανάληψης Π.Ε. 16 Μαρτίου 2021
Ημερομηνία περάτωσης Π.Ε. 19 Ιουνίου 2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Ο κατηγοριοποιητής εγγυτέρων γειτόνων είναι ένας από τους πιο γνωστούς κατηγοριοποιητές, καθώς είναι απλός και εύκολος στην εφαρμογή του παρουσιάζει όμως μερικά μειονεκτήματα. Ένα από αυτά είναι το είναι το υψηλό υπολογιστικό κόστος σε περιπτώσεις μεγάλων συνόλων δεδομένων το οποίο καθιστά τη χρήση του κατηγοριοποιητή αναποτελεσματική. Για την αντιμετώπιση αυτού του μειονεκτήματος μπορεί να χρησιμοποιηθεί είτε κάποια μέθοδος δεικτοδότησης πολυδιάστατων δεδομένων είτε κάποια τεχνική μείωση των δεδομένων εκπαίδευσης. Οι μέθοδοι δεικτοδότησης δημιουργούν μια δομή δεδομένων που μπορεί να μειώσουν σημαντικά το πλήθος των αποστάσεων που υπολογίζονται για την εύρεση και την ανάκτηση των εγγύτερων γειτόνων, παρουσιάζουν τεράστια προβλήματα όμως όταν το σύνολο δεδομένων περιέχει πολλά χαρακτηριστικά. Αντίστοιχα, οι τεχνικές μείωσης δεδομένων προ-επεξεργάζονται τα δεδομένα και δημιουργούν ένα μικρό σύνολο δεδομένων εκπαίδευσης που έχει ως στόχο τη μείωση του υπολογιστικού κόστους και ταυτόχρονα, διατήρηση της ακρίβειας σε υψηλά επίπεδα.

Η βιβλιοθήκη scikit-learn της Python, προσφέρει υλοποιήσεις δυο γνωστών μεθόδους δεικτοδότησης. Αυτές είναι το kd-tree και το ball-tree. Στόχος την παρούσας πτυχιακής εργασίας είναι η εκπόνηση μια εκτεταμένης πειραματικής μελέτης στο περιβάλλον του scikit-learn της Python όπου οι μέθοδοι δεικτοδότησης θα συγκρίνονται με τις τεχνικές μείωσης δεδομένων χρησιμοποιώντας πολλά διαφορετικά σύνολα δεδομένων με διαφορετικό πλήθος διαστάσεων το κάθε ένα. Κατά την πειραματική μελέτη εκτιμήθηκε σαν μέτρο απόδοσης τόσο η ακρίβεια κατηγοριοποίησης όσο και ο χρόνος αναζήτησης εγγυτέρων γειτόνων.

Εξετάστηκαν δώδεκα σύνολα δεδομένων και μέσα από τα αποτελέσματα του πειράματος βγαίνει το συμπέρασμα ότι, όταν ένα σύνολο δεδομένων έχει παραπάνω από 10 χαρακτηριστικά είναι πιο αποδοτικό να χρησιμοποιήσουμε κάποια τεχνική μείωσης δεδομένων.

Multidimensional data indexing methods and data reduction techniques for effective nearest neighbor classification

Dimitrios Manolis

Abstract

The nearest neighbor classifier is one of the most well-known classifiers, as it is simple and easy to implement but has some disadvantages. One of these is the high computational cost in cases of large datasets which makes the use of the classifier inefficient. To address this disadvantage, either a method of multidimensional data indexing or a data reduction technique of the training data can be used. Indexing methods create a data structure that can significantly reduce the number of calculations to find and retrieve the closest neighbors, but present huge problems when the dataset has many features. Correspondingly, data reduction techniques pre-process the data and create a small training dataset that aims to reduce computational costs and at the same time, maintain accuracy at high levels.

Python's scikit-learn library offers implementations of two well-known indexing methods. These are kd-tree and ball-tree. The aim of this paper is to conduct an extensive experimental study in the scikit-learn environment of Python where indexing methods will be compared with data reduction techniques using many different datasets with different number of features each. In the experimental study, both the accuracy of classification and the search time of the nearest neighbors are assessed as a measure of efficiency.

Twelve data sets were examined and the results of the experiment suggest that when a data set has more than 10 features it is more efficient to use a data reduction technique.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Κατηγοριοποίηση

Η κατηγοριοποίηση ή εποπτευόμενη μάθηση σύμφωνα με την ορολογία της μηχανικής μάθησης είναι ένα κρίσιμο έργο της εξόρυξης δεδομένων. Οι αλγόριθμοι κατηγοριοποίησης (ή κατηγοριοποιητές) [7] προσπαθούν να αντιστοιχίσουν νέα, μη κατηγοριοποιημένα αντικείμενα σε ένα σύνολο προκαθορισμένων κλάσεων, με βάση τα διαθέσιμα δεδομένα εκπαίδευσης, δηλαδή ένα σύνολο αντικειμένων τα οποία είναι ήδη κατηγοριοποιημένα. Ένα τυπικό παράδειγμα κατηγοριοποίησης είναι να αντιστοιχίσουμε ένα μήνυμα ηλεκτρονικού ταχυδρομείου είτε στην κλάση "spam" είτε στην κλάση "non-spam" ή να προβλέψουμε αν θα ρίξει χαλάζι ή όχι σε μια δεδομένη χρονική στιγμή.

Οι κατηγοριοποιητές μπορούν να χωριστούν σε δύο κύριες κατηγορίες αλγορίθμων [7]: πρώτον στους πρόθυμους κατηγοριοποιητές (eager classifiers) και δεύτερον τους οκνηρούς κατηγοριοποιητές (ή με βάση παραδειγμάτων κατηγοριοποιητές). Και οι δύο έχουν το ίδιο κίνητρο, δηλαδή την ακριβής πρόβλεψη της σωστής κλάσης ενός νέου αντικειμένου. Ωστόσο, διαφέρουν ως προς τον τρόπο λειτουργίας τους. Βασικό ρόλο για την αποτελεσματικότητα των αλγορίθμων και των δύο κατηγοριών παίζει το διαθέσιμο σύνολο εκπαίδευσης. Ένας πρόθυμος κατηγοριοποιητής προ-επεξεργάζεται τα διαθέσιμα δεδομένα εκπαίδευσης πριν από την κατηγοριοποίηση και δημιουργεί ένα μοντέλο κατηγοριοποίησης που στη συνέχεια χρησιμοποιείται για την κατηγοριοποίηση νέων, μη κατηγοριοποιημένων αντικειμένων. Από την άλλη πλευρά, οι οκνηροί κατηγοριοποιητές δεν κατασκευάζουν κανένα μοντέλο κατηγοριοποίησης. Στην πραγματικότητα, θεωρούν το σύνολο δεδομένων εκπαίδευσης ως μοντέλο κατηγοριοποίησης. Ένας οκνηρός αλγόριθμος κατηγοριοποιεί ένα νέο αντικείμενο τη στιγμή που φτάνει σε αυτόν σαρώνοντας το σύνολο εκπαίδευσης.

Δεδομένου ότι οι πρόθυμοι κατηγοριοποιητές δημιουργούν ένα μοντέλο κατηγοριοποίησης πριν φτάσει σε αυτούς οποιοδήποτε νέο αντικείμενο, η διαδικασία κατηγοριοποίησης είναι πολύ γρήγορη. Αν και οι οκνηροί κατηγοριοποιητές δεν ξοδεύουν χρόνο για να δημιουργήσουν μοντέλα κατηγοριοποίησης, η διαδικασία κατηγοριοποίησης τους είναι πιο χρονοβόρα από αυτή των προθύμων κατηγοριοποιητών. Ένα μειονέκτημα των πρόθυμων κατηγοριοποιητών είναι ότι πρέπει να δημιουργήσουν μια ενιαία υπόθεση που να καλύπτει ολόκληρο το σύνολο εκπαίδευσης. Αυτό όμως δεν είναι πάντα εφικτό, έτσι μπορεί να επηρεάσει την ακρίβεια κατηγοριοποίησης και μπορεί να καταστήσει την κατασκευή του μοντέλου κατηγοριοποίησης μια εξαιρετικά χρονοβόρα και πολύπλοκη διαδικασία προ-επεξεργασίας. Από την άλλη πλευρά, οι οκνηροί κατηγοριοποιητές χρησιμοποιούν ολόκληρο το σύνολο εκπαίδευσης και ως εκ τούτου, μπορούν να υιοθετήσουν πιο περίπλοκες υποθέσεις σχετικά με τα δεδομένα. Κατά συνέπεια, μπορούν να αποδείξουν την ακρίβεια της κατηγοριοποίησης . Ένα μειονέκτημα των οκνηρών κατηγοριοποιητών είναι ότι απαιτούν όλα τα δεδομένα εκπαίδευσης να είναι πάντα διαθέσιμα, γεγονός που

οδηγεί σε υψηλές απαιτήσεις αποθήκευσης των δεδομένων. Αντίθετα, πρόθυμοι κατηγοριοποιητές, όταν κατασκευάσουν το μοντέλο κατηγοριοποίησης, δεν χρειάζονται πια τα δεδομένα εκπαίδευσης και έτσι μπορούν να αφαιρεθούν για να εξοικονομηθεί χώρος αποθήκευσης.

Κατά τη διάρκεια των τελευταίων δεκαετιών, και με την αύξησή της δημοτικότητας της μηχανικής μάθησης και της εξόρυξης δεδομένων το πρόβλημα της κατηγοριοποίησης έχει προκαλέσει το ενδιαφέρον πολλών ερευνητών από διαφορετικούς ερευνητικούς τομείς της επιστήμης των υπολογιστών. Ως εκ τούτου, έχουν προταθεί και είναι διαθέσιμοι διάφοροι πρόθυμοι και οκνηροί κατηγοριοποιητές.

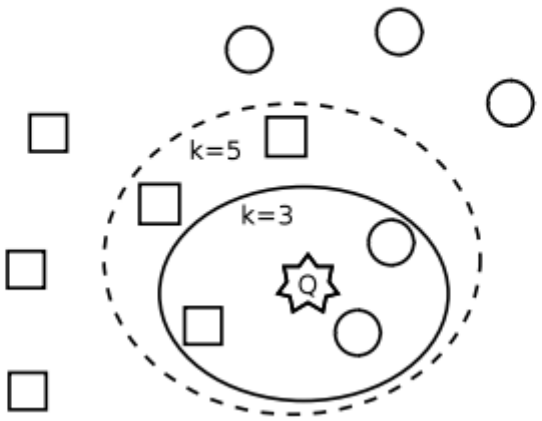
Τα δέντρα αποφάσεων κατηγοριοποίησης [8] αποτελούν μια πολύ γνωστή υποκατηγορία πρόθυμων κατηγοριοποιητών. Με βάση τα διαθέσιμα δεδομένα εκπαίδευσης, αυτοί οι κατηγοριοποιητές δημιουργούν μια δενδρική δομή που χρησιμοποιείται για την κατηγοριοποίηση των νέων αντικειμένων. Άλλοι πρόθυμοι κατηγοριοποιητές βασίζονται σε τεχνητά νευρωνικά δίκτυα [9]. Ένα νευρωνικό δίκτυο εκπαιδεύεται πρώτα από τα αντικείμενα εκπαίδευσης και στη συνέχεια εκτελεί κατηγοριοποιήσεις. Στην κατηγορία των προθύμων κατηγοριοποιητών ανήκουν επίσης και οι πιθανοτικοι κατηγοριοποιητές. Αυτοί δημιουργούν ένα μοντέλο κατηγοριοποίησης που βασίζεται σε πιθανότητες. Ένα χαρακτηριστικό παράδειγμα ενός πιθανοτικού κατηγοριοποιητή είναι ο naive Bayes [10]. Μια άλλη υποκατηγορία αλγορίθμων κατηγοριοποίησης περιλαμβάνει τους κατηγοριοποιητές που βασίζονται σε κανόνες συσχέτισης [11], αυτοί ανακαλύπτουν κανόνες συσχέτισης μέσα στα διαθέσιμα δεδομένα εκπαίδευσης. Έπειτα αυτοί οι κανόνες χρησιμοποιούνται για την κατηγοριοποίηση.

Από την άλλη πλευρά, η κατηγορία των οκνηρών κατηγοριοποιητών περιλαμβάνει τον γνωστό κατηγοριοποιητή k εγγύτερων γειτόνων [12, 13] και τις μεθόδους κατηγοριοποίησης με βάση την υπόθεση (case-based reasoning) [14] . Ο κατηγοριοποιητής k εγγύτερων γειτόνων είναι το αντικείμενο μελέτης της παρούσας πτυχιακής εργασίας.

1.2 Κατηγοριοποιητής k εγγυτέρων γειτόνων

Ο κατηγοριοποιητής k εγγύτερων γειτόνων (k-NN) [12, 13] είναι ένας αποτελεσματικός και με εκτεταμένη χρήση οκνηρός κατηγοριοποιητής. Είναι απλός και εύκολος στην εφαρμογή, και μπορεί να αξιοποιηθεί σε πολλούς τομείς εφαρμογών και εύκολα να ενσωματωθεί σε πολλά συστήματα. Επιπλέον, ο κατηγοριοποιητης k-NN είναι αναλυτικά ανιχνεύσιμος [12].

Δεδομένου ότι ο κατηγοριοποιητής k-NN είναι ένας οκνηρός κατηγοριοποιητής , δεν δημιουργεί κανένα μοντέλο κατηγοριοποίησης. Ο αλγόριθμος χρησιμοποιεί τα δεδομένα εκπαίδευσης κάθε φορά που χρειάζεται να κατηγοριοποιηθεί ένα νέο αντικείμενο. Συγκεκριμένα, κατηγοριοποιεί ένα αντικείμενο X αναζητώντας στα διαθέσιμα δεδομένα εκπαίδευσης και ανακτώντας τα k εγγύτερα αντικείμενα (γείτονες) στο X σύμφωνα με μια μέτρηση απόστασης. Στη συνέχεια, στο X αποδίδεται η ετικέτα της πιο κοινής κλάση μεταξύ των ετικετών των ανακτηθέντων



Σχήμα 1.1: Κατηγοριοποιητής k-NN για k=3 και k=5 [1]

k εγγυτέρων γειτόνων. Αυτή η κλάση ονομάζεται συχνά η κύρια κλάση και καθορίζεται μέσω μιας διαδικασίας γνωστής ως ψηφοφορία εγγυτέρων γειτόνων (nearest neighbours voting). Όταν η τιμή k = 1, ο αλγόριθμος είναι επίσης γνωστός και ως κατηγοριοποιητής ενός εγγύτερου γείτονα (ή κανόνας 1-NN)

Το σχήμα 1.1 απεικονίζει ένα παράδειγμα της διαδικασίας κατηγοριοποίησης σε δυο διαστάσεις. Πιο συγκεκριμένα, απεικονίζει ένα σύνολο δεδομένων που αποτελείτε από δύο κλάσεις, τα τετράγωνα και τους κύκλους και ένα αντικείμενο ερωτήματος (Q) που πρέπει να κατηγοριοποιηθεί σε μία από αυτές τις δύο κλάσεις. Αν ορίσουμε το k να είναι ίσο με τρία (κύκλος χωρίς διακεκομμένες γραμμές στο σχήμα 1.1), το Q κατηγοριοποιείται στην κλάση κύκλος επειδή δύο από τους τρεις εγγύτερους γείτονες είναι κύκλοι. Από την άλλη πλευρά, εάν το k έχει οριστεί να είναι ίσο με πέντε (κύκλος με διακεκομμένες γραμμές), το αντικείμενο ερωτήματος κατηγοριοποιείται στη κλάση τετράγωνο επειδή τρεις από τους πέντε εγγύτερους γείτονες ανήκουν στη κλάση τετράγωνο.

Η απόδοση της κατηγοριοποίησης ασφαλώς και εξαρτάται από την τιμής της παραμέτρου k. Η τιμή του k που επιτυγχάνει την υψηλότερη ακρίβεια κατηγοριοποίησης εξαρτάται από το σύνολο δεδομένων που χρησιμοποιούμε κάθε φορά και ο προσδιορισμός της συνήθως συνεπάγεται με τον συντονισμό μέσω δαπανηρών διεργασιών προ-επεξεργασίας δοκιμής και σφάλματος. Αν και ο προσδιορισμός του k δεν μπορεί να ακολουθήσει κανέναν γενικό κανόνα γιατί το καλύτερο k μπορεί να είναι εντελώς διαφορετικό για διαφορετικά σύνολα δεδομένων. Οι μεγαλύτερες τιμές k είναι καταλληλότερες για σύνολα δεδομένων που περιέχουν θόρυβο, δεδομένου ότι εξετάζουν μεγαλύτερες γειτονιές και έτσι δεν επηρεάζονται από ένα ή δυο αντικείμενα που μπορεί να δείχνουν σε λάθος κλάση. Ωστόσο, δεν καθορίζουν σαφώς τα όρια μεταξύ ξεχωριστών κλάσεων. Αντίθετα, οι μικρές τιμές παραμέτρων καθιστούν τον κατηγοριοποιητή πιο ευαίσθητο στον θόρυβο (π,χ με k=1 τα αντικείμενά δίπλα στο αντικείμενο θόρυβο θα είναι πάντα λάθος

κατηγοριοποιημένα). Επομένως, σε περιπτώσεις δεδομένων εκπαίδευσης που περιέχουν θόρυβο, η κατηγοριοποίηση είναι πιθανώς λιγότερο ακριβής.

Αξίζει να σημειωθεί ότι ακόμη και η καλύτερη τιμή k μπορεί να μην είναι η βέλτιστη. Αυτό συμβαίνει επειδή ο κατηγοριοποιητής k-NN χρησιμοποιεί μια μοναδική τιμή k. Διαφορετικές τιμές k μπορεί να είναι βέλτιστες για διαφορετικές περιοχές του συνόλου δεδομένων. Π.χ. σε μια γειτονιά του συνόλου που περιέχει αρκετό θόρυβο μπορεί η βέλτιστη τιμή του k να είναι το 7, ενώ σε μια άλλη γειτονιά του ίδιου συνόλου που δεν έχει πολλά αντικείμενά και καθόλου θόρυβο η βέλτιστη τιμή μπορεί να είναι το 3. Συνεπώς, μπορούν να υιοθετηθούν ευρετικές μέθοδοι για τον δυναμικό προσδιορισμό του k [15] που μπορεί να επιτυγχάνει και υψηλότερη ακρίβεια από τον κατηγοριοποιητή k-NN που έχει ως παράμετρο την "καλύτερη" τιμή k.

Σε περιπτώσεις προβλημάτων δυαδικής κατηγοριοποίησης (σύνολα δεδομένων που έχουν μόνο δύο κλάσεις), το k θα πρέπει να έχει μια περιττή τιμή για να αποφύγει τις ισοπαλίες κλάσεων κατά τη διάρκεια της ψηφοφορίας εγγύτερων γειτόνων. Σε περιπτώσεις μη δυαδικών προβλημάτων, το k μπορεί να έχει οποιαδήποτε τιμή. Σε αυτήν την περίπτωση, οι πιθανές ισοπαλίες κατά τη διάρκεια της ψηφοφορίας επιλύονται επιλέγοντας είτε μια τυχαία "πιο κοινή" κλάση είτε την κλάση του πλησιέστερου στο αντικείμενο γείτονα. Το δημοφιλές λογισμικό Weka [16] και πολλά άλλα εργαλεία λογισμικού εξόρυξης δεδομένων / μηχανικής μάθησης επιλύουν τυχαία τις ισοπαλίες.

Ένα άλλο σημαντικό ζήτημα που θα πρέπει να αντιμετωπιστεί είναι η επιλογή της μετρικής που χρησιμοποιείται για τον υπολογισμό της απόστασης μεταξύ των αντικειμένων. Βεβαίως, η απόφαση αυτή θα πρέπει να λαμβάνει υπόψη τους τύπους δεδομένων των χαρακτηριστικών του συνόλου δεδομένων (μεταβλητές). Σε περιπτώσεις πραγματικών ή και ακέραιων χαρακτηριστικών, η ευκλείδεια απόσταση είναι η συνήθως η χρησιμοποιούμενη μέτρηση απόστασης. Ωστόσο, μπορούν να υιοθετηθούν και άλλες μετρήσεις απόστασης (π.χ. Minkowski, Mahalanobis,Chebyshev,Manhattan,Hamming) [17]. Πολλά άλλα μέτρα ομοιότητας έχουν προταθεί για τον χειρισμό ονομαστικών χαρακτηριστικών (μη μετρικοί χώροι). Ωστόσο, σε αυτήν την πτυχιακή εργασία όλα τα σύνολα δεδομένων έχουν πραγματικά ή και ακέραια χαρακτηριστικά. Επομένως, θα χρησιμοποιήσουμε την Ευκλείδεια απόσταση ως μετρική. Συνεπώς, τα αντικείμενα δεδομένων που περιγράφονται από n χαρακτηριστικά θεωρούνται ως σημεία δεδομένων (ή διανύσματα) στον Ευκλείδειο μετρικό χώρο n-διαστάσεων και η ευκλείδεια απόσταση μεταξύ των σημείων p και q δίνεται από τον παρακάτω τύπο 1.1:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \qquad (1.1)$$

Τα διαφορετικά εύρη χαρακτηριστικών μπορούν να επηρεάσουν την τιμή απόστασης. Ακόμη

και σε περιπτώσεις όπου όλα τα χαρακτηριστικά έχουν την ίδια σημασία, τα χαρακτηριστικά με μεγαλύτερα εύρη τιμων έχουν μεγαλύτερη επίδραση στην τιμή απόστασης από τα χαρακτηριστικά με μικρότερα εύρη τιμών. Ας υποθέσουμε ότι το χαρακτηριστικό " μισθός" κυμαίνεται από 800 έως 5000 και το χαρακτηριστικό "ηλικία " παίρνει τιμές από 1 έως 100. Αν και τα δύο χαρακτηριστικά έχουν την ίδια σημασία, ο "μισθός" έχει μεγαλύτερο αντίκτυπο στον υπολογισμό απόστασης από την "ηλικία". Επομένως, το εύρος των χαρακτηριστικών θα πρέπει να κανονικοποιηθεί σε ένα συγκεκριμένο εύρος διαστήματος (π.χ. [0, 1]). Ας υποθέσουμε ότι ένα σύνολο δεδομένων περιέχει n αντικείμενα, ένα χαρακτηριστικό του συνόλου ας το πούμε e θα πρέπει να είναι κανονικοποιημένο σε [0, 1]. η τιμή e του *i*-στου αντικείμενου, *i* = 1, . . .,*n* κανονικοποιείται ως εξής:

$$normalized(e_i) = \frac{e_i \ - \ Emin}{Emax \ - \ Emin} \tag{1.2}$$

όπου $Emin$ και $Emax$ είναι αντίστοιχα οι ελάχιστες και μέγιστες τιμές για το χαρακτηριστικό e. Η κανονικοποίηση δεδομένων είναι μια κοινή διαδικασία προ-επεξεργασίας σε πολλές εργασίες εξόρυξης δεδομένων. Ορισμένα λογισμικά εξόρυξης δεδομένων έχουν προεπιλεγμένη την εφαρμογή τους. Σε αυτήν την πτυχιακή εργασία η κανονικοποίησης αναφέρεται και ως κλιμάκωση (scaling) γιατί έτσι αναφέρεται στο scikit-learn.

Έχουν προταθεί διάφορες παραλλαγές του κατηγοριοποιητή k-NN. Ο σημαντικότερος είναι ο κατηγοριοποιητής k-NN με βαρύτητα αποστάσεων [18], ο οποίος χρησιμοποιεί μια συνάρτηση απόστασης-βάρους για να βαρύνει περισσότερο τους εγγύτερους γείτονες από τους πιο απομακρυσμένους. Ο πλησιέστερος γείτονας έχει βάρος κοντά στο ένα ενώ ο πιο απομακρυσμένος γείτονας έχει βάρος κοντά στο μηδέν. Τα βάρη όλων των άλλων γειτόνων κλιμακώνονται σε αυτό το διάστημα. Ένα νέο αντικείμενο κατηγοριοποιείται με ψηφοφορία βάση των βαρών, δηλαδή αποδίδεται στην κλάση με το μεγαλύτερο άθροισμα των βαρών. Ο συγκεκριμένος κατηγοριοποιητης προσφέρεται από το scikit-learn.

Άλλη μια παραλλαγή του κατηγοριοποιητή k-NN είναι ο κατηγοριοποιητης "γείτονες μέσα σε μια ακτίνα (Radius Neighbors Classifier)" [19] . Ο χρήστης ορίζει μια προεπιλεγμένη τιμή r και έτσι η κλάση του αντικείμενου αποφασίζεται μόνο από γείτονες μέσα σε αυτήν την ακτίνα. Με αυτόν τον τρόπο σε πιο αραιές γειτονιές δεν παίρνουμε υπόψη πολύ μακρινούς γείτονες. Και αυτός ο κατηγοριοποιητής προσφέρεται από το scikit-learn.

1.3 Μειονεκτήματα k-εγγυτέρων γειτόνων

Αν και ο κατηγοριοποιητής k-NN θεωρείται ένας αποτελεσματικός αλγόριθμος, έχει κάποιες αδυναμίες που μπορεί να καταστήσουν τη χρήση του αναποτελεσματική. Το πρώτο αδύνατο σημείο του, είναι το υψηλό υπολογιστικό κόστος. Ο κατηγοριοποιητής k-NN πρέπει να υπολογίζει όλες τις αποστάσεις μεταξύ κάθε μη κατηγοριοποιημένου αντικειμένου και όλων των

αντικειμένων που είναι αποθηκευμένα στο σύνολο εκπαίδευσης. Σε περιπτώσεις μεγάλων συνόλων δεδομένων, αυτό το μειονέκτημα καθιστά τη χρήση του μια χρονοβόρα και σε ορισμένες περιπτώσεις απαγορευτική διαδικασία. Για παράδειγμα, υποθέστε ότι ένα σύστημα κατηγοριοποίησης αποθηκεύει 400.000 αντικείμενα εκπαίδευσης. Επιπλέον, υποθέστε ότι το σύστημα θα πρέπει να κατηγοριοποιήσει περίπου 100.000 μη κατηγοριοποιημενα αντικείμενα εκτελώντας τον κατηγοριοποιητή k-NN πάνω από τα δεδομένα εκπαίδευσης. Αυτό σημαίνει ότι το σύστημα πρέπει να υπολογίσει σαράντα δισεκατομμύρια αποστάσεις. Αν και σήμερα τα συστήματα είναι εξοπλισμένα με ισχυρούς επεξεργαστές, αυτοί οι υπολογισμοί είναι χρονοβόροι και απαράδεκτοι. Αυτή η αδυναμία μπορεί να αντιμετωπιστεί με την χρήση κάποιας μεθόδου δεικτοδότησης πολυδιάστατων δεδομένων (multi-attribute indexing) ή με την χρήση κάποιας τεχνικής μείωσης δεδομένων (data reduction technique).

Πρέπει να αναφέρουμε ότι, εκτός από το μέγεθος του συνόλου εκπαίδευσης, το υπολογιστικό κόστος της διαδικασίας της κατηγοριοποίησης εξαρτάται επίσης από τον αριθμό των διαστάσεων των δεδομένων. Όσες περισσότερες είναι οι διαστάσεις των δεδομένων, τόσο πιο πολλοί υπολογισμοί χρειάζονται να εκτελεστούν για έναν υπολογισμό απόστασης. Αυτό επηρεάζει την απόδοση του k-NN, άλλα καθιστά την χρήση των μεθόδων δεικτοδότησης αναποτελεσματική, όσο μεγαλώνει ο αριθμός των χαρακτηριστικών ενός αντικειμένου.

Μια άλλη αδυναμία του κατηγοριοποιητή k-NN είναι οι μεγάλες απαιτήσεις αποθήκευσης για τα δεδομένα εκπαίδευσης. Σε αντίθεση με τους πρόθυμους κατηγοριοποιητές που μπορούν να διαγράφουν από τη μνήμη τα δεδομένα εκπαίδευσης μετά την κατασκευή του μοντέλου κατηγοριοποίησης , ο κατηγοριοποιητής k-NN χρειάζεται τα δεδομένα εκπαίδευσης να είναι πάντα διαθέσιμα. Κατά συνέπεια, ο κατηγοριοποιητής k-NN πρέπει να εκτελείται σε συστήματα υπολογιστών εξοπλισμένα με αρκετή κύρια μνήμη για την αποθήκευση των δεδομένων εκπαίδευσης.

Η τελευταία αδυναμία του κατηγοριοποιητή k-NN είναι ότι, όπως και πολλές άλλες μέθοδοι κατηγοριοποίησης, είναι μια μέθοδος ευαίσθητη στο θόρυβο. Συγκεκριμένα, η ακρίβεια κατηγοριοποίησης εξαρτάται σε μεγάλο βαθμό από την ποιότητα των δεδομένων εκπαίδευσης. Ο θόρυβος και τα λανθασμένα δεδομένα, καθώς και οι ακραίες τιμές και οι επικαλύψεις μεταξύ περιοχών δεδομένων διαφορετικών κλάσεων, οδηγούν σε λιγότερο ακριβή κατηγοριοποίηση. Η χρήση υψηλών τιμών k επεκτείνει την εξεταζόμενη γειτονιά και, ως εκ τούτου, μπορεί να διορθώσει εν μέρει αυτό το μειονέκτημα. Ωστόσο, συνεπάγεται υψηλό αριθμό εκτελέσεων δοκιμής και σφάλματος για τον προσδιορισμό της κατάλληλης τιμής k.

Αυτές οι αδυναμίες αποτελούν ενεργό ερευνητικό πρόβλημα και έχουν προσελκύσει το ενδιαφέρον της ερευνητικής κοινότητας του κλάδου της εξόρυξης δεδομένων. Αυτή η πτυχιακή εργασία βασίζεται επίσης σε κάποιες από αυτές τις αδυναμίες. Πιο συγκεκριμένα στις πρώτες δυο που αναφέραμε, δηλαδή στο υψηλό υπολογιστικό κόστος και την μείωση της απόδοσης όσο αυξάνονται οι διαστάσεις.

Έχουν προταθεί πολυάριθμοι αλγόριθμοι και τεχνικές που μπορούν να αντιμετωπίσουν τα αδύνατα σημεία του κατηγοριοποιητή. Μια πιθανή κατηγοριοποίηση των μεθόδων για βελτιωμένη κατηγοριοποίηση k-NN είναι: (i) δεικτοδότηση πολυδιάστατων δεδομένων , (ii) και τεχνικές μείωσης δεδομένων.

Η δεικτοδότηση πολυδιάστατων δεδομένων [2, 20, 21] μπορεί να επιταχύνει τις αναζητήσεις εγγυτέρων γειτόνων. Ωστόσο, οι απαιτήσεις αποθήκευσης αυξάνονται, καθώς εκτός από τα δεδομένα εκπαίδευσης, πρέπει να αποθηκευτεί και ο δείκτης. Επιπλέον, οι δείκτες μπορούν να εφαρμοστούν μόνο σε σύνολα δεδομένων με σχετικά μικρές διαστάσεις (π.χ. 2-19) λόγο του φαινομένου της κατάρας των διαστάσεων [22].

Οι τεχνικές μείωσης δεδομένων (DRTs) χωρίζονται σε δυο μεγάλες κατηγορίες: (i) τη μείωση αντικειμένων και, (ii) τη μείωση διαστάσεων. Σε αυτήν την πτυχιακή εργασία αν και θα δούμε και τις δυο κατηγορίες, θα ασχοληθούμε κυρίως με την πρώτη. Οι ΤΜΔ μπορούν να αντιμετωπίσουν αποτελεσματικά όλες τις αδυναμίες και μπορούν να ομαδοποιηθούν σε δύο κύριες κατηγορίες αλγορίθμων: (i) αλγόριθμοι επιλογής προτύπων [23] και (ii) αλγόριθμοι αφαίρεσης (ή παραγωγής) προτύπων [24]. Οι αλγόριθμοι επιλογής προτύπων επιλέγουν αντιπροσωπευτικά αντικείμενα (ή πρωτότυπα) από το αρχικό σύνολο εκπαίδευσης, ενώ οι αλγόριθμοι αφαίρεσης προτύπων δημιουργούν αντικείμενο συνοψίζοντας σε παρόμοια αντικείμενα εκπαίδευσης και τα χρησιμοποιούν ως πρωτότυπα. Στην πραγματικότητα, κάθε πρωτότυπο αντιπροσωπεύει μια συγκεκριμένη περιοχή δεδομένων του πολυδιάστατου χώρου.

1.4 Κίνητρο και Συνεισφορά

Όπως αναφέραμε ένα σημαντικό μειονέκτημα του κατηγοριοποιητή εγγυτέρων γειτόνων είναι το υπολογιστικό του κόστος. Έτσι έχουν εφευρεθεί διάφορες μέθοδοι για να καταπολεμήσουν αυτό το μειονέκτημα. Στην παρούσα πτυχιακή εργασία μελετάμε τις μεθόδους δεικτοδότησης πολυδιάστατων δεδομένων και τις τεχνικές μείωσης δεδομένων. Οι μέθοδοί δεικτοδότησης αν και πάρα πολύ αποτελεσματικοί σε σύνολα χαμηλών διαστάσεων/χαρακτηριστικών, πέφτουν σε απόδοση όταν εξετάζουν ένα σύνολο με πολλά χαρακτηριστικά ενώ είναι τελείως αναποτελεσματικοί σε σύνολα με περισσότερα από ένα όριο χαρακτηριστικών εξαιτίας της λεγομένης κατάρας των διαστάσεων.

Αρκετές έρευνές έχουν γίνει για να συγκρίνουν διάφορες μεθόδους δεικτοδότησης μεταξύ τους και άλλες έρευνες για να συγκρίνουν τις ΤΜΔ μεταξύ τους. Σκοπός της παρούσας πτυχιακής εργασίας είναι να συγκρίνουμε τις διαθέσιμες μεθόδους δεικτοδότησης της δημοφιλούς βιβλιοθήκης scikit-learn και τις πιο διαδεδομένες ΤΜΔ που είναι διαθέσιμες. Για να γίνει αυτό θα πρέπει να πραγματοποιήσουμε μια πειραματική μελέτη μέσω της γλωσσάς προγραμματισμού Python στην οποία θα συγκρίνουμε τις παραπάνω μεθόδους χρησιμοποιώντας πολλά διαφορετικά σύνολα δεδομένων με διαφορετικό πλήθος διαστάσεων το κάθε ένα. Στα πλαίσια της πειραματικής μελέτης θα εκτιμηθεί τόσο η ακρίβεια στην κατηγοριοποίηση όσο και οι χρόνοι

CPU που απαιτούνται για να την αναζήτηση των εγγύτερων γειτόνων. Μέσα από την πειραματική μελέτη θα πρέπει να καταλήξουμε σε ένα συμπέρασμα, ποιος από τους δυο τρόπους είναι είναι καταλληλότερος για αναζήτηση εγγυτέρων γειτόνων σε ένα σύνολο με δεδομένο αριθμό χαρακτηριστικών.

1.5 Οργάνωση της Πτυχιακής

Στο δεύτερο κεφάλαιο αναλύονται οι δυο τρόποι που μπορούν να επιταχύνουν τις αναζητήσεις εγγυτέρων γειτόνων, δηλαδή μέθοδοί δεικτοδότησης πολυδιάστατων δεδομένων και τεχνικές μείωσης δεδομένων. Πρώτα αναλύονται οι μέθοδοι δεικτοδότησης, η έννοια της κατάρας των διαστάσεων και της εγγενής διάστασης και οι μέθοδοι που χρησιμοποιούνται στην πειραματική ανάλυση δηλαδή kd-tree και ball-tree. Έπειτα αναλύονται λεπτομερώς οι τεχνικές μείωσης δεδομένων που χρησιμοποιούνται, δηλαδή ENN-rule, CNN-rule, IB2 και RSP3.

Στο τρίτο κεφάλαιο παρουσιάζεται η γλώσσα Python και η βιβλιοθήκη εξόρυξης δεδομένων scikit-learn, μέσα από την οποία πραγματοποιούνται τα πειράματα. Έπειτα αναλύονται έννοιες όπως ακρίβεια κατηγοριοποίησης και οι μέθοδοι αποτίμησης της, π.χ το cross validation, καθώς και οι κλάσεις και οι συναρτήσεις του scikit-learn που είναι αναγκαίες για τα πειράματα μας.

Το τέταρτο κεφάλαιο είναι η πειραματική ανάλυση. Πρώτα παρουσιάζονται τα σύνολα δεδομένων που χρησιμοποιούνται. Έπειτα παρουσιάζονται οι παράμετροι του πειράματος και οι μετρήσεις του πρώτου πειράματος. Μετά συγκρίνονται τα αποτελέσματα από τις μετρήσεις για κάθε σύνολο ξεχωριστά. Έπειτα πραγματοποιείται το δεύτερο πείραμα για το πως η κατάρα των διαστάσεων επηρεάζει τις μεθόδους δεικτοδότησης.

Τέλος, στο πέμπτο κεφάλαιο παρουσιάζεται το τελικό συμπέρασμα που σχηματίστηκε μέσω της πειραματικής ανάλυσης που εκτελέστηκε στην παρούσα πτυχιακή εργασία ενώ παράλληλα παρουσιάζονται κατευθύνσεις για μελλοντική έρευνα.

Κεφάλαιο 2ο: Τεχνικές και αλγόριθμοι για γρήγορη κατηγοριοποίηση εγγύτερων γειτόνων

2.1 Εισαγωγή

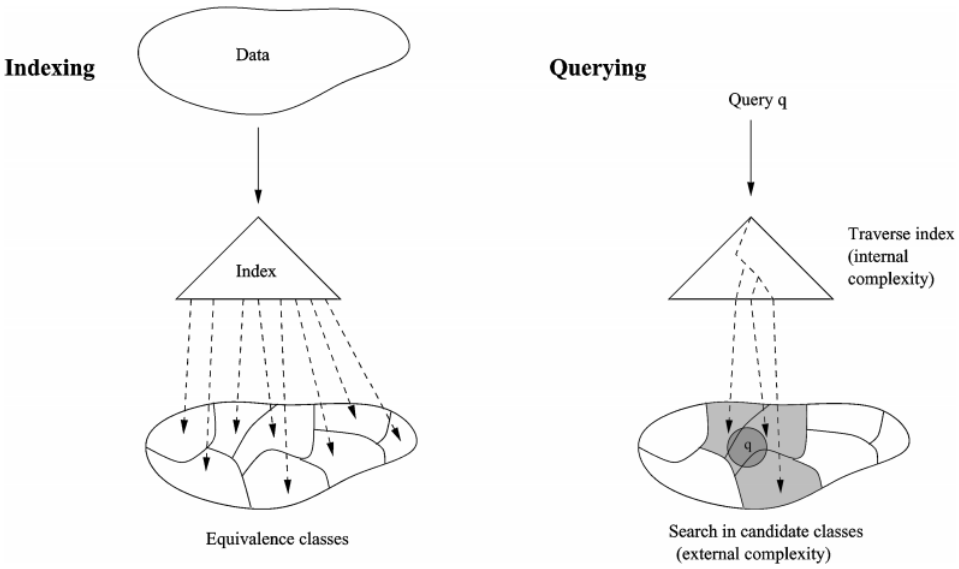
Σε αυτό το κεφάλαιο θα επικεντρωθούμε στις μεθόδους που έχουν ήδη παρουσιαστεί λεπτομερώς από διάφορους ερευνητές και χρησιμοποιούνται για στην εκπόνηση της παρούσας εργασίας. Το κεφάλαιο έχει χωριστεί σε δύο μέρη οπού το κάθε μέρος αναπτύσσεται και χωρίζεται περαιτέρω.

Το πρώτο μέρος είναι η Ενότητα 2.2, όπου παρουσιάζεται η έννοια των μεθόδων δεικτοδότησης. Σε αυτήν την ενότητα θα εξηγήσουμε τι είναι η δεικτοδότηση πολυδιάστατων δεδομένων, η κατάρα των διαστάσεων, η εγγενής διάσταση καθώς και θα μελετήσουμε αναλυτικά τις δυο μεθόδους δεικτοδότησης που είναι διαθέσιμες από την βιβλιοθήκη scikit-learn και χρησιμοποιούνται για την συγκεκριμένη εργασία, δηλαδή το kd-tree και το ball-tree και πως αυτά μπορούν να επιταχύνουν την διαδικασία των k εγγυτέρων γειτόνων.

Το δεύτερο μέρος είναι η Ενότητα 2.3, η οποία αφορά στις Τεχνικές Μείωσης Δεδομένων. Αρχίζουμε με μία ανάλυση των διαθέσιμων κατηγοριών ΤΜΔ που υπάρχουν και έπειτα παρουσιάζονται οι τεχνικές που χρησιμοποιήθηκαν στα πειράματα της παρούσας εργασίας, δηλαδή Edited Nearest Neighbor - rule, Condensing Nearest Neighbor - rule, Instance-Based Learning και Reduction by Space Partioning.

2.2 Δεικτοδότηση πολυδιάστατων δεδομένων

Ένας αλγόριθμος δεικτοδότησης ή ευρετηρίασης (indexing) είναι μια διαδικασία εκτός σύνδεσης, δηλαδή ένας αλγόριθμος που λειτουργεί σε ολόκληρο το σύνολο δεδομένων ταυτόχρονα και χρειάζεται όλα τα δεδομένα κατά την λειτουργία του για την κατασκευή μιας δομής δεδομένων που ονομάζεται δείκτης ή ευρετήριο, η οποία έχει σχεδιαστεί για να αποθηκεύει υπολογισμούς απόστασης έτσι ώστε να απαντά σε ερωτήματα εγγύτητας (όπως ο αλγόριθμος k-NN) αργότερα. Οι μέθοδοί δεικτοδότησης απαιτούν την προ-επεξεργασία του συνόλου εκπαίδευσης ώστε να κατασκευαστεί η μορφή του δείκτη. Αυτές οι μέθοδοι χρησιμοποιούνται κυρίως σε βάσεις δεδομένων ώστε να αυξήσουμε την ταχύτητα των ερωτημάτων. Αυτή η δομή δεδομένων μπορεί να είναι δαπανηρή κατά την κατασκευή της, αλλά αυτό αποσβένεται με την εξοικονόμηση υπολογισμών αποστάσεων σε πολλά ερωτήματα. Ως εκ τούτου, ο στόχος είναι ο σχεδιασμός αποτελεσματικών αλγορίθμων δεικτοδότησης για τη μείωση του αριθμού των υπολογισμών αποστάσεων. Όλες αυτές οι δομές λειτουργούν με βάση την απόρριψη αντικειμένων χρησιμοποιώντας την τριγωνική ανισότητα (η μόνη ιδιότητα που επιτρέπει την εξοικονόμηση αξιολογήσεων απόστασης) $|x + y| \leq |x| + |y|$.



Σχήμα 2.1: Το ενοποιημένο μοντέλο για την δεικτοδότηση και την αναζήτηση μετρικών χώρων [2]

Σε αυτήν την πτυχιακή εργασία χρησιμοποιούμε μεθόδους δεικτοδότησης πολυδιάστατων δεδομένων. Γενικά μπορούμε να αναφερθούμε σε αυτές, και ως μεθόδους δεικτοδότησης του μετρικού χώρου (metric space indexing) γιατί θεωρούμε ότι τα πολυδιάστατα δεδομένα υπάρχουν στον μετρικό χώρο. Σε ορισμένες περίπτωσης όπως η δίκη μας, ο μετρικός χώρος αποδεικνύεται ότι είναι ενός συγκεκριμένου τύπου που ονομάζεται “διανυσματικός χώρος (vector space)”, όπου τα αντικείμενα αποτελούνται από k συντεταγμένες πραγματικών τιμών, άρα πιο συγκεκριμένα μέθοδοί όπως το kd-tree, που είναι από τις πιο γνωστές μεθόδους δεικτοδότησης και αντικείμενο αυτής της πτυχιακής εργασίας, θεωρούμε ότι δεικτοδοτούν τον διανυσματικό χώρο [2]. Ένας διανυσματικός χώρος επιτρέπει περισσότερη ελευθερία από έναν γενικό μετρικό χώρο κατά το σχεδιασμό αναζητήσεων εγγύτητας, καθώς είναι δυνατή η χρήση γεωμετρικών πληροφοριών και συντεταγμένων που δεν είναι διαθέσιμες σε ένα γενικό μετρικό χώρο [25].

Το πρόβλημά είναι πως μπορούμε να χωρίσουμε τον χώρο n-διαστάσεων ώστε να δημιουργήσουμε το δείκτη. Χρειαζόμαστε λοιπόν και μια μέθοδο για να αποφασίζει πως θα χωρίζεται. Οι σημαντικότερες αυτές μέθοδοι βασίζονται σε δομές δεδομένων δέντρων, κάποιες από αυτές τις μεθόδους είναι το k-dimensional-tree (kd-tree), το k-dimensional-B-tree (kd-B-tree), το Vantage Point-tree (VP-tree), το Ball-tree και το R-tree.

Ένα μειονέκτημα είναι ότι, οι δείκτες αυξάνουν τις απαιτήσεις αποθήκευσης. Τα δεδομένα εκπαίδευσης πρέπει να είναι πάντα διαθέσιμα και ο δείκτης πρέπει να αποθηκεύεται επίσης. Επιπλέον, τέτοιες μέθοδοι μπορούν να εφαρμοστούν μόνο σε σύνολα δεδομένων με μέτρια διάσταση (π.χ. 2-19). Σε υψηλότερες διαστάσεις, η χρήση της δεικτοδότησης δεν έχει νόημα. Το φαινόμενο της “κατάρας των διαστάσεων” καθιστά τους δείκτες άχρηστους, καθώς η απόδοσή τους υποβαθμίζεται γρήγορα και μπορεί να γίνει χειρότερη από αυτή της εξαντλητικής

αναζήτησης ολόκληρου του συνόλου δεδομένων [22]. Πολλές προτάσεις για την επιτάχυνση των αναζητήσεων εγγυτέρων γειτόνων βασίζονται στη μείωση των διαστάσεων προκειμένου να εφαρμοστεί αποτελεσματικά [26]. Ωστόσο, αυτό μπορεί συχνά να οδηγήσει σε σημαντική απώλεια πληροφοριών. Επιπλέον, κάθε μη κατηγοριοποιημένο αντικείμενο πρέπει να μετασχηματιστεί πριν από την αναζήτηση.

2.2.1 Η Κατάρα των διαστάσεων και εγγενής διάσταση

Ένα από τα σημαντικότερα εμπόδια για το σχεδιασμό αποτελεσματικών τεχνικών αναζήτησης σε μετρικούς χώρους είναι η ύπαρξη και παρουσία των λεγόμενων χώρων υψηλής διάστασης σε πραγματικές εφαρμογές. Οι παραδοσιακές τεχνικές δεικτοδότησης για διανυσματικούς χώρους (όπως τα kd-trees) έχουν μια εκθετική εξάρτηση από την διάσταση του χώρου (καθώς ο όγκος ενός τετραγώνου ή ενός υπερκύβου που περιέχει τις απαντήσεις αυξάνεται εκθετικά με τη διάσταση) [2, 27].

Πιο πρόσφατες τεχνικές δεικτοδότησης για διανυσματικούς χώρους και εκείνες για γενικούς μετρικούς χώρους μπορούν να απαλλαγούν από την αναπαράσταση της διάσταση του χώρου. Αυτό κάνει μεγάλη διαφορά σε πολλές εφαρμογές που χειρίζονται διανυσματικούς χώρους υψηλών διαστάσεων αλλά χαμηλής εγγενούς διάστασης (Intrinsic dimension), (π.χ ένα επίπεδο βυθισμένο σε ένα διανυσματικό χώρο 50 διαστάσεων ή απλά συσταδοποιημένα δεδομένα). Ωστόσο, σε ορισμένες περιπτώσεις ακόμη και η εγγενής διάσταση είναι πολύ υψηλή και το πρόβλημα καθίσταται δυσεπίλυτο για ακριβείς αλγόριθμους (όπως ο k-NN) και πρέπει να καταφύγουμε σε προσεγγιστικούς ή πιθανοτικούς αλγόριθμους.

Η εγγενής διάσταση για ένα σύνολο δεδομένων μπορεί να θεωρηθεί ως ο ελάχιστος αριθμός των μεταβλητών που απαιτούνται για την αναπαράσταση των δεδομένων [28]. Ομοίως, στην επεξεργασία σήματος πολυδιάστατων σημάτων, η εγγενής διάσταση του σήματος περιγράφει πόσες μεταβλητές χρειάζονται για να δημιουργηθεί μια καλή προσέγγιση του σήματος.

Οι διανυσματικοί χώροι μπορεί να υποφέρουν από μεγάλες διαφορές μεταξύ της αναπαράστασης της διάστασης k και της εγγενούς τους διάστασης. Για παράδειγμα, ένα επίπεδο ενσωματωμένο σε ένα χώρο 50 διαστάσεων έχει εγγενή διάσταση 2 άλλα η αναπαράσταση της διάστασης του είναι 50. Αυτό είναι γενικά, η περίπτωση των πραγματικών εφαρμογών όπου τα δεδομένα είναι συσταδοποιημένα, και έχει οδηγήσει σε προσπάθειες μέτρησης της εγγενούς διάστασης όπως η έννοια της μορφοκλασματικής διάστασης (fractal dimension). Παρά το γεγονός ότι καμία τεχνική δεν μπορεί να αντιμετωπίσει εγγενή διάσταση μεγαλύτερη από 20, πολύ υψηλότερες αναπαραστάσεις διαστάσεων μπορούν να αντιμετωπιστούν με τεχνικές μείωσης διαστάσεων [29].

Κατά την εκτίμηση της εγγενούς διάστασης, ωστόσο, χρησιμοποιείται συχνά ένας ευρύτερος ορισμός που βασίζεται στην πολλαπλή διάσταση, όπου μια αναπαράσταση στην εγγενή διάσταση χρειάζεται μόνο να υπάρχει τοπικά. Τέτοιες μέθοδοι εκτίμησης εγγενών διαστάσεων μπορούν

έτσι να χειριστούν σύνολα δεδομένων με διαφορετικές εγγενείς διαστάσεις σε διαφορετικά μέρη του συνόλου δεδομένων. Αυτό συχνά αναφέρεται ως τοπική εγγενής διάσταση [28].

Η εγγενής διάσταση μπορεί να χρησιμοποιηθεί ως κατώτερο όριο για μέχρι ποια διάσταση είναι δυνατή η συμπίεση ενός συνόλου δεδομένων μέσω της μείωσης των διαστάσεων, αλλά μπορεί επίσης να χρησιμοποιηθεί ως μέτρο της πολυπλοκότητας του συνόλου δεδομένων. Για ένα σύνολο δεδομένων N διαστάσεων , η εγγενής διάσταση M ικανοποιείται από την σχέση $0 \leq M \leq N$.

2.2.2 KD-Tree

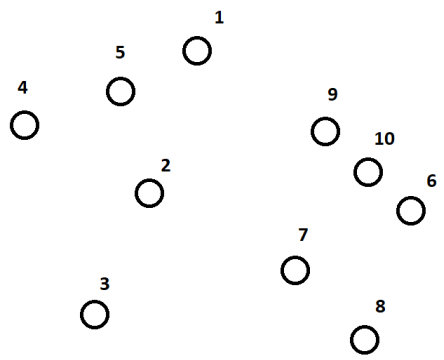
Τα δέντρα k-διαστάσεων (k-dimensional tree, kd-tree) είναι μια δενδρική δομή δεδομένων διαχωρισμού του χώρου για την οργάνωση σημείων στο χώρο k διαστάσεων που εφευρέθηκε από τον Jon Bentley [30,31] . Τα kd-trees είναι χρήσιμες δομές για εφαρμογές όπως αναζητήσεις που περιλαμβάνουν ένα πολυδιάστατο κλειδί αναζήτησης π.χ οι αναζήτηση εγγυτέρων γειτόνων.

Το kd-tree θεωρείτε ένα δυαδικό δέντρο στο οποίο κάθε κόμβος φύλλο (leaf node) είναι ένα σημείο k διαστάσεων. Κάθε κόμβος που δεν είναι φύλλο μπορεί να θεωρηθεί ότι δημιουργεί ένα υπερεπίπεδο διαχωρισμού που χωρίζει το χώρο σε δύο μέρη, γνωστά ως μισά διαστήματα. Τα αντικείμενα στα αριστερά αυτού του υπερεπιπέδου αντιπροσωπεύονται από το αριστερό υποδέντρο αυτού του κόμβου και τα αντικείμενα στα δεξιά του υπερεπιπέδου αντιπροσωπεύονται από το δεξιό υποδέντρο [30].

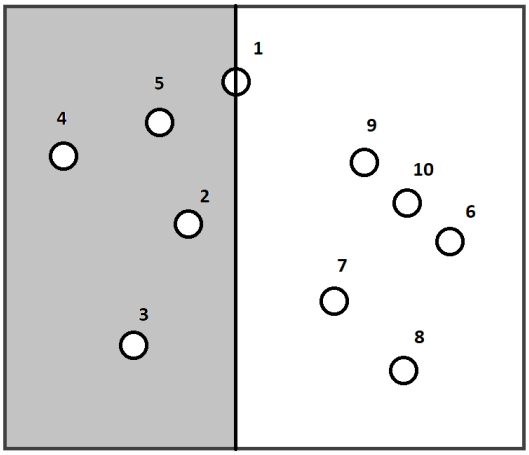
Η κατεύθυνση του υπερεπιπέδου επιλέγεται με τον ακόλουθο τρόπο: κάθε κόμβος στο δέντρο συνδέεται με μία από τις k διαστάσεις, με το υπερεπίπεδο κάθετο στον άξονα αυτής της διάστασης. Έτσι, για παράδειγμα, αν για μια συγκεκριμένη διαίρεση του χώρου επιλέγεται ο άξονας “y”, όλα τα αντικείμενα στο υποδέντρο με μικρότερη τιμή “y” από τον κόμβο θα εμφανιστούν στο αριστερό υποδέντρο και όλα τα αντικείμενα με μεγαλύτερη τιμή “y” θα βρίσκονται στο δεξιό υποδέντρο.

Κατά την κατασκευή της απλής εκδοχής του kd-tree [32] καθώς κάποιος κινείται προς την κάτω μεριά του δέντρου, εναλλάσσουμε κυκλικά τους άξονες που χρησιμοποιούνται για την επιλογή των επιπέδων διαχωρισμού. Σε ένα τρισδιάστατο δέντρο, η ρίζα θα είχε ένα επίπεδο ευθυγραμμισμένο με το X, τα παιδιά της ρίζας θα έχουν και τα δύο, επίπεδο ευθυγραμμισμένο με το Y , τα εγγόνια της ρίζα θα έχουν όλα, το επίπεδο τους ευθυγραμμισμένο με το Z, τα δισέγγονα της ρίζας θα έχουν όλα, επίπεδο ευθυγραμμισμένο με το X, τα τρισέγγονα της ρίζας θα έχουν όλα, επίπεδο ευθυγραμμισμένο με το Y και ούτω καθεξής.

Τα αντικείμενα εισάγονται επιλέγοντας το διάμεσο των αντικείμενων που υπάρχουν στο υποδέντρο σε σχέση με τις συντεταγμένες τους στον άξονα που χρησιμοποιείται για τη δημιουργία του επιπέδου διαχωρισμού (τροφοδοτούμε ολόκληρο το σύνολο των N αντικειμένων στον αλγόριθμο εκ των προτέρων). Αυτή η μέθοδος οδηγεί σε ένα ισορροπημένο k-d tree, στο οποίο κάθε κόμβος φύλλο έχει περίπου την ίδια απόσταση από τη ρίζα. Ωστόσο, τα ισορροπημένα

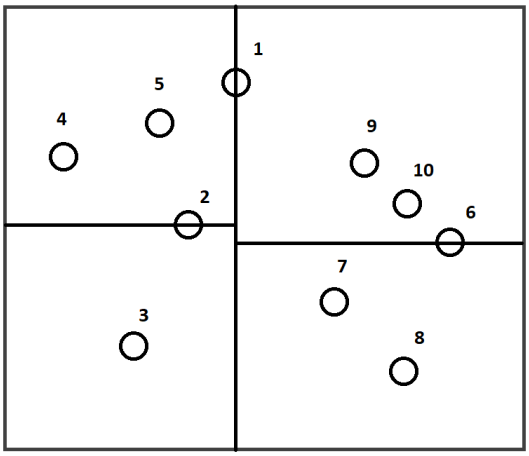


2.2.1. Το σύνολο δεδομένων

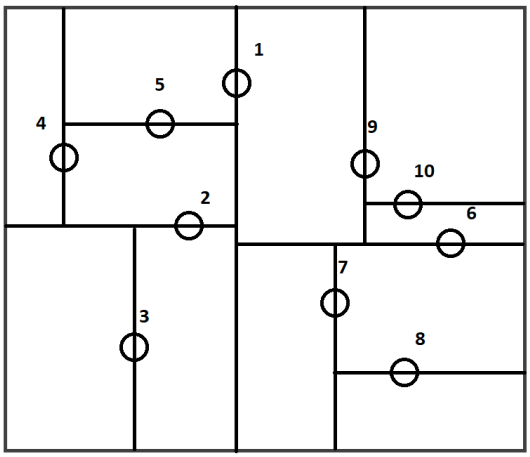


2.2.2. Πρώτος διαχωρισμός

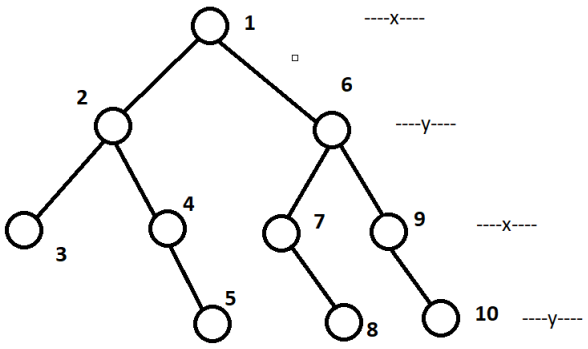
c



2.2.3. Δεύτερος διαχωρισμός



2.2.4. Τελευταίος διαχωρισμός



2.2.5. Τελική μορφή του kd-tree

Σχήμα 2.2: Κατασκευή ενός kd-tree

δέντρα δεν είναι απαραίτητα τα βέλτιστα για όλες τις εφαρμογές.

Στο σχήμα 2.2 παρουσιάζεται ένα παράδειγμα κατασκευής kd-tree σε δυο διαστάσεις. Στο σχήμα 2.2.1 φαίνεται το σύνολο δεδομένων,το οποίο περιέχει 10 αντικείμενα, από το οποίο θα κατασκευαστεί το δέντρο. Στο σχήμα 2.2.2 παρουσιάζεται ο πρώτος διαχωρισμός, οποίος γίνεται στο αντικείμενο 1 και στον άξονα x. Όπως φαίνεται στο σχήμα 2.2.3 ο άξονα του διαχωρισμού αλλάζει κυκλικά και έτσι ο δεύτερος διαχωρισμός συμβαίνει στον άξονα y. Στο σχήμα 2.2.4 παρουσιάζεται ο τελευταίος διαχωρισμός του συνόλου δεδομένων. Και τέλος στο σχήμα 2.2.5 παρουσιάζεται το δέντρο το οποίο θα κατασκευαστεί μέσα από τους διαχωρισμούς.

Δεν απαιτείται η επιλογή του διάμεσου σημείου. Εάν δεν έχουν επιλεγεί διάμεσα σημεία, δεν υπάρχει εγγύηση ότι το το δέντρο θα είναι ισορροπημένο. Για να αποφεύγουμε να φτιάξουμε έναν αλγόριθμο εύρεσής διάμεσου με πολυπλοκότητα O(n) ή να χρησιμοποιήσουμε ένα O(n log n) ταξινομητή όπως ο heapsort ή ο mergesort για να ταξινομήσουμε όλα τα αντικείμενα N [33], μια δημοφιλής πρακτική είναι να ταξινομήσουμε με έναν προεπιλεγμένο σταθερό αριθμό τυχαίων επιλεγμένων σημείων και να χρησιμοποιήσουμε τη διάμεση τιμή αυτών των σημείων ως το επίπεδο διάσπασης. Στην πράξη, αυτή η τεχνική συχνά έχει ως αποτέλεσμα ομοιόμορφα ισορροπημένα δέντρα.

Όταν θέλουμε να προσθέσουμε ένα νέο αντικείμενο σε ένα k-d tree το κάνουμε με τον ίδιο τρόπο που κάποιος προσθέτει ένα αντικείμενο σε οποιοδήποτε άλλο δέντρο αναζήτησης. Αρχικά, διασχίζουμε το δέντρο, ξεκινώντας από τη ρίζα και μετακινώντας είτε προς τα αριστερά είτε προς τα δεξιά, ανάλογα με το αν το αντικείμενο που πρόκειται να εισαχθεί βρίσκεται στην “αριστερή” ή “δεξιά” πλευρά του επιπέδου διαχωρισμού. Μόλις φτάσουμε στον κόμβο κάτω από τον οποίο πρέπει να βρίσκεται το νέο αντικείμενο, προσθέτουμε το νέο αντικείμενο είτε ως αριστερό είτε ως δεξί παιδί του κόμβου φύλλου, και πάλι ανάλογα με το ποια πλευρά του επιπέδου διαχωρισμού του κόμβου περιέχει τον νέο κόμβο.

Η προσθήκη αντικειμένων με αυτόν τον τρόπο μπορεί να προκαλέσει την ανισορροπία του δέντρου, οδηγώντας στην μειωμένη απόδοση του δέντρου. Ο ρυθμός υποβάθμισης της απόδοσης των δένδρων εξαρτάται από τη χωρική κατανομή των αντικειμένων που προστίθενται στο δέντρο και από τον αριθμό των αντικειμένων που προστίθενται σε σχέση με το μέγεθος του δέντρου. Εάν ένα δέντρο γίνει μη ισορροπημένο, ίσως χρειαστεί να εξισορροπηθεί για να αποκατασταθεί η απόδοση των ερωτημάτων που βασίζονται στην εξισορρόπηση του δέντρου, όπως η αναζήτηση εγγύτερου γείτονα.

Για να αφαιρέσουμε ένα αντικείμενο από ένα υπάρχον k-d tree, ο ευκολότερος τρόπος είναι να δημιουργήσουμε ένα σύνολο όλων των κόμβων και των φύλλων από τα παιδιά του κόμβου που σκοπεύουμε να διαγράψουμε και να ξαναδημιουργήσουμε αυτό το τμήμα του δέντρου.

Μια άλλη προσέγγιση είναι να βρεθεί ένας αντικαταστάτης για το αντικείμενο που αφαιρέθηκε. Πρώτον, βρίσκουμε τον κόμβο R που περιέχει το αντικείμενο που πρέπει να αφαιρεθεί. Για την βασική περίπτωση όπου το R είναι κόμβος φύλλο, δεν απαιτείται αντικατάσταση. Για τη γενική περίπτωση οπού ο R δεν είναι κόμβος φύλλο, βρίσκουμε ένα αντικείμενο αντικατάστασης, ας

πούμε p, από το υποδέντρο που έχει ρίζα τον κόμβο R. Αντικαθίσταται το αντικείμενο που είναι αποθηκευμένο στο R με το p. Στη συνέχεια, καταργείται αναδρομικά το p.

Για την εύρεση ενός αντικείμενου αντικατάστασης, εάν το R κάνει την διάκριση του ας πούμε στον άξονα και το R έχει ένα παιδί στα δεξιά, βρίσκουμε το αντικείμενο με την ελάχιστη τιμή X από το υποδέντρο που έχει ρίζα στο δεξιό παιδί του R. Διαφορετικά, βρίσκουμε το αντικείμενο με τη μέγιστη τιμή X από το υποδέντρο που έχει ρίζα το αριστερό παιδί του αντικείμενου R.

Ο αλγόριθμος αναζήτησης εγγύτερου γείτονα στοχεύει να βρει το αντικείμενο στο δέντρο που βρίσκεται πλησιέστερα σε ένα δεδομένο αντικείμενο εισόδου. Αυτή η αναζήτηση μπορεί να γίνει αποτελεσματικά χρησιμοποιώντας τις ιδιότητες δέντρου για να εξαλείψει γρήγορα μεγάλα τμήματα του χώρου αναζήτησης.

Η αναζήτηση ενός εγγύτερου γείτονα σε ένα k-d tree προχωρά ως εξής:

1. Ξεκινώντας από τον κόμβο ρίζας, ο αλγόριθμος κινείται προς τα κάτω στο δέντρο αναδρομικά, με τον ίδιο τρόπο που θα κατέβαινε αν το αντικείμενο αναζήτησης ήθελε να εισαχθεί στο δέντρο (δηλαδή πηγαίνει αριστερά ή δεξιά ανάλογα με το αν το αντικείμενο είναι μικρότερο ή μεγαλύτερο από τον τρέχοντα κόμβο στη διάσταση διαίρεσης).

2. Μόλις ο αλγόριθμος φτάσει σε έναν κόμβο φύλλο, ελέγχει αυτό το αντικείμενο κόμβου και αυτό το αντικείμενο αποθηκεύεται ως το τρέχον καλύτερο.

3. Ο αλγόριθμος εκτυλίσσει την αναδρομή του δέντρου, εκτελώντας τα ακόλουθα βήματα σε κάθε κόμβο:

    i Εάν ο τρέχων κόμβος είναι πιο κοντά από τον τρέχοντα καλύτερο, τότε γίνεται ο τρέχων καλύτερος.

    ii Ο αλγόριθμος ελέγχει αν θα μπορούσαν να υπάρχουν αντικείμενα στην άλλη πλευρά του επιπέδου διαχωρισμού που είναι πιο κοντά στο αντικείμενο αναζήτησης από το τρέχον καλύτερο. Αυτό γίνεται με τη τομή του υπερεπιπέδου διαίρεσης με μια υπερσφαίρα γύρω από το αντικείμενο αναζήτησης που έχει ακτίνα ίση με την τρέχουσα εγγύτερη απόσταση. Δεδομένου ότι τα υπερεπίπεδα είναι όλα ευθυγραμμισμένα με τους άξονες, αυτό εφαρμόζεται ως μια απλή σύγκριση για να δούμε αν η απόσταση μεταξύ του υπερεπιπέδου διαίρεσης και του αντικείμενου αναζήτησης είναι μικρότερη από την απόσταση από το αντικείμενο αναζήτησης στο τρέχον καλύτερο.

        - Εάν η υπερσφαίρα διασχίσει το επίπεδο, θα μπορούσαν να υπάρχουν εγγύτερα αντικείμενα από το τρέχων καλύτερο στην άλλη πλευρά του επιπέδου, οπότε ο αλγόριθμος πρέπει να μετακινηθεί προς τα κάτω στον άλλο κλάδο του δέντρου από τον τρέχοντα κόμβο αναζητώντας εγγύτερα αντικείμενα, ακολουθώντας την ίδια αναδρομική διαδικασία με ολόκληρη την αναζήτηση.

- Εάν η υπερσφαίρα δεν τέμνει το επίπεδο διαχωρισμού , τότε ο αλγόριθμος συνεχίζει να περπατάει πάνω στο δέντρο και εξαλείφεται ολόκληρος το κλαδί του δέντρου στην άλλη πλευρά αυτού του κόμβου.

4. Όταν ο αλγόριθμος ολοκληρώσει αυτή τη διαδικασία για τον κόμβο ρίζας, τότε η αναζήτηση ολοκληρώνεται.

Γενικά ο αλγόριθμος χρησιμοποιεί τετραγωνικές αποστάσεις για σύγκριση για να αποφύγει τον υπολογισμό τετραγωνικών ριζών. Επιπλέον, μπορεί να μειώσει το υπολογιστικό κόστος κρατώντας την τετραγωνισμένη τρέχουσα καλύτερη απόσταση σε μια μεταβλητή για σύγκριση [34].

Ο αλγόριθμος μπορεί να επεκταθεί με διάφορους τρόπους και με απλές τροποποιήσεις. Μπορεί να παρέχει τους k εγγυτέρους γείτονες σε ένα αντικείμενο διατηρώντας τα k τρέχοντα καλύτερα αντί για ένα μόνο. Ένα κλαδί εξαλείφεται μόνο όταν έχουν βρεθεί k αντικείμενα και το κλαδί δεν μπορεί να έχει αντικείμενα πιο κοντά από οποιοδήποτε από τα k τρέχοντα καλύτερα , και στην παρών πτυχιακή εργασία αυτή η επέκταση του αλγορίθμου είναι αυτή που χρειαζόμαστε. Ο αλγόριθμος μπορεί επίσης και να κατασκευαστεί έτσι ώστε τα αντικείμενα να είναι αποθηκευμένα μόνο στους κόμβους φύλλα [32].

Μπορεί επίσης να μετατραπεί σε έναν αλγόριθμο προσέγγισης για να τρέξει πιο γρήγορα [35]. Για παράδειγμα, η κατά προσέγγιση αναζήτηση εγγύτερου γείτονα μπορεί να επιτευχθεί απλά θέτοντας ένα ανώτερο όριο για τον αριθμό των αντικειμένων που εξετάζονται στο στο δέντρο ή διακόπτοντας τη διαδικασία αναζήτησης με βάση ένα ρολόι σε πραγματικό χρόνο (η οποία μπορεί να είναι πιο κατάλληλη σε υλοποιήσεις υλικού). Ο εγγύτερος γείτονας για τα αντικείμενα που βρίσκονται στο δέντρο ήδη μπορεί να επιτευχθεί με την μη ενημέρωση της βελτίωσης για κόμβους που δίνουν μηδενική απόσταση, ως αποτέλεσμα αυτό έχει το μειονέκτημα της απόρριψης αντικειμένων που δεν είναι μοναδικά, αλλά βρίσκονται μαζί με το αρχικό αντικείμενο αναζήτησης. Ο κατά προσέγγιση εγγύτερος γείτονας είναι χρήσιμος σε εφαρμογές σε πραγματικό χρόνο, όπως η ρομποτική, λόγω της σημαντικής αύξησης της ταχύτητας που αποκτάται από το να μην ψάχνουμε εξαντλητικά το καλύτερο αντικείμενο.

Όταν υπάρχουν πολλές διαστάσεις, η κατάρα των διαστάσεων αναγκάζει τον αλγόριθμο να χρειάζεται να επισκεφθεί πολλούς περισσότερους κλάδους από ό, τι θα επισκεπτόταν αν είχαμε λίγες διαστάσεις. Κατά γενικό κανόνα, εάν η διάσταση είναι k, ο αριθμός των αντικειμένων n, πρέπει να είναι $n >> 2^k$ [30]. Διαφορετικά, όταν τα k-d trees χρησιμοποιούνται με δεδομένα υψηλών διαστάσεων, τα περισσότερα από τα αντικείμενα του δέντρου θα αξιολογηθούν και η αποτελεσματικότητα δεν θα είναι καλύτερη από την εξαντλητική αναζήτηση [36] και, εάν απαιτείται μια αρκετά γρήγορη απάντηση, θα πρέπει να χρησιμοποιηθούν μέθοδοί κατά προσέγγιση εγγύτερου γείτονα.

2.2.3 Ball-Tree

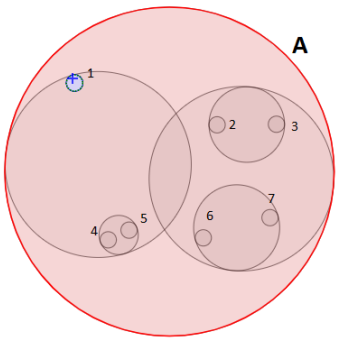
Στην επιστήμη των υπολογιστών, το ball-tree [37] είναι μια δομή δεδομένων διαχωρισμού του χώρου για την οργάνωση σημείων σε ένα πολυδιάστατο χώρο. Το ball-tree παίρνει το όνομα του από το γεγονός ότι χωρίζει τα σημεία δεδομένων σε ένα ένθετο σύνολο υπερσφαιριδίων D-διαστάσεων γνωστών ως "μπάλες". Η προκύπτουσα δομή δεδομένων έχει χαρακτηριστικά που την καθιστούν χρήσιμη για μια σειρά εφαρμογών, κυρίως την εφαρμογή αναζήτησης εγγύτερου γείτονα.

Το ball-tree είναι ένα δυαδικό δέντρο στο οποίο κάθε κόμβος ορίζει μια D-διαστάσεων υπερσφαίρα, η οποία περιέχει ένα υποσύνολο των αντικειμένων που πρέπει να αναζητηθούν. Οι εσωτερικοί κόμβοι χρησιμοποιούνται μόνο για να καθοδηγήσουν την αποτελεσματική αναζήτηση μέχρι τα φύλλα του δέντρου. Κάθε εσωτερικός κόμβος του δέντρου χωρίζει τα αντικείμενα δεδομένων σε δύο ξεχωριστά σύνολα τα οποία συνδέονται με διαφορετικές μπάλες. Τα φύλλα του δέντρου κατέχουν τις σχετικές με την εφαρμογή πληροφορίες. Κάθε κόμβος φύλλο στο δέντρο ορίζει μια μπάλα και απαριθμεί όλα τα αντικείμενα δεδομένων μέσα σε αυτήν την μπάλα.

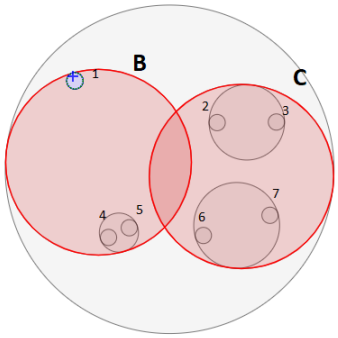
Οι μπάλες αντιπροσωπεύονται από τις n + 1 τιμές κινητής υποδιαστολής που καθορίζουν τις συντεταγμένες του κέντρου τους και το μήκος της ακτίνας τους. Μια μπάλα συνδέεται με ένα κόμβο με τέτοιο τρόπο, ώστε η μπάλα ενός εσωτερικού κόμβου να είναι η μικρότερη δυνατή μπάλα, η οποία περιέχει τα παιδιά του. Σε αντίθεση με τις περιοχές κόμβων σε kd-trees , οι αδελφικές περιοχές στα ball trees επιτρέπεται να τέμνονται και δεν απαιτείται να χωρίσουν ολόκληρο το χώρο. Ενώ οι ίδιες οι μπάλες μπορεί να τέμνονται, κάθε αντικείμενο αποδίδεται στην μία ή την άλλη μπάλα κατά το διαχωρισμό ανάλογα με την απόστασή της από το κέντρο της μπάλας.

Μια σειρά από αλγόριθμους κατασκευής ενός ball-tree είναι διαθέσιμοι. Ο στόχος ενός τέτοιου αλγορίθμου είναι να παράγει ένα δέντρο που θα υποστηρίζει αποτελεσματικά ερωτήματα του επιθυμητού τύπου (π.χ. εγγύτερου γείτονα) στη μέση περίπτωση. Τα συγκεκριμένα κριτήρια ενός ιδανικού δέντρου εξαρτώνται από τον τύπο της ερώτησης που απαντάται και την κατανομή των δεδομένων. Ωστόσο, ένα γενικά εφαρμόσιμο μέτρο ενός αποτελεσματικού δέντρου είναι αυτό που ελαχιστοποιεί τον συνολικό όγκο των εσωτερικών κόμβων του. Δεδομένου των ποικίλων κατανομών των συνόλων δεδομένων πραγματικού κόσμου, αυτό είναι ένα δύσκολο έργο, αλλά υπάρχουν πολλές ευρετικές μέθοδοι που χωρίζουν τα δεδομένα στην πράξη. Γενικά, υπάρχει μια ανταλλαγή μεταξύ του κόστους κατασκευής ενός δέντρου και της αποτελεσματικότητας που επιτυγχάνεται με αυτή τη μέτρηση [38].

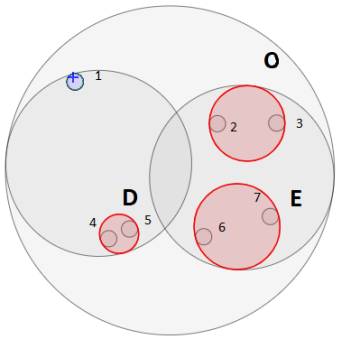
Παρακάτω περιγράφεται ο απλούστερος από αυτούς τους αλγόριθμους, τον οποίο και χρησιμοποιεί το scikit-learn. Το scikit-learn μας παραπέμπει σε μια πιο εμπεριστατωμένη έρευνα πέντε αλγορίθμων, μαζί με αυτόν που θα δούμε, που δόθηκε από τον Stephen Omohundro [37]. Ο οποίος κατασκεύασε τα ball trees από ήδη έτοιμα φύλλα κόμβους. Η δομή των εσωτερικών μπαλών καθορίζεται από κάτω προς τα πάνω από τις μπάλες των παιδιών του.



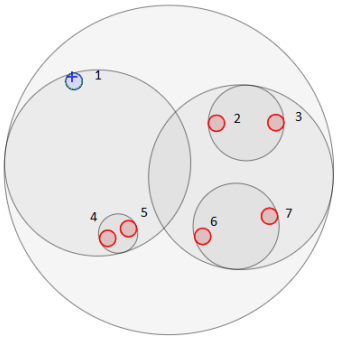
2.3.1. Βάθος 0



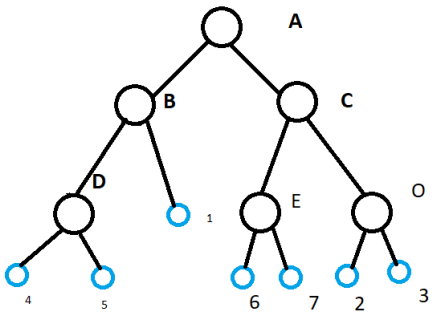
2.3.2. Βάθος 1



2.3.3. Βάθος 2



2.3.4. Βάθος 3



2.3.5. Τελική μορφή του ball-tree

Σχήμα 2.3: Ball-tree κατασκευασμένο με αλγόριθμο kd [3]

Η απλούστερη τέτοια διαδικασία ονομάζεται “αλγόριθμος κατασκευής kd”, και είναι ανάλογή με τη διαδικασία που χρησιμοποιείται για την κατασκευή kd-tree. Το δέντρο είναι χτισμένο από πάνω προς τα κάτω με αναδρομικό διαχωρισμό των δεδομένων σε δύο υποσύνολα. Σε κάθε επανάληψη ο διαχωρισμός γίνεται κατά μήκος της διάστασης με τη μεγαλύτερη εξάπλωση των αντικειμένων, με το σύνολο να χωρίζεται από τη διάμεση τιμή όλων των αντικειμένων κατά μήκος αυτής της διάστασης. Η διάμετρος κάθε εσωτερικού κόμβου είναι η μέγιστη απόσταση μεταξύ των παιδιών του. Στο σχήμα 2.3 παρουσιάζεται ένα ball-tree κατασκευασμένο με τον

αλγόριθμο kd.

Σε μια πραγματική εφαρμογή όμως δεν μπορούν να υπάρχουν έτοιμα φύλλα κόμβοι. Για αυτόν τον λόγο πρέπει να ξέρουμε ποτέ θα πρέπει να σταματήσει η ανάδρομη, έτσι θέτουμε ένα διάστημα τιμών πχ 30-60 ώστε τόσα να είναι τα αντικείμενα ενός κόμβου για να θεωρείται κόμβος φύλλο, η διάμετρος κάθε κόμβου φύλλο καθορίζεται από την απόσταση των δυο πιο απομακρυσμένων αντικείμενων μέσα σε αυτόν.

Μια σημαντική εφαρμογή των ball-trees είναι η επιτάχυνση των ερωτημάτων αναζήτησης εγγύτερων γειτόνων [39], στα οποία ο στόχος είναι να βρεθούν τα k αντικείμενα στο δέντρο που είναι πιο κοντά σε ένα δεδομένο αντικείμενο ερωτήματος με κάποια μέτρηση απόστασης. Ένας απλός αλγόριθμος αναζήτησης,εκμεταλλεύεται την ιδιότητα αποστάσεων του ball-tree. Συγκεκριμένα, εάν ο αλγόριθμος κάνει αναζήτησή στο ball tree με ένα αντικείμενο ερωτήματος T και έχει ήδη εντοπίσει κάποιο αντικείμενο P που είναι το εγγύτερο στο T από τα αντικείμενα που έχουν συναντηθεί μέχρι στιγμής, τότε κάθε υποδέντρο του οποίου η μπάλα απέχει περισσότερο από το T από ότι το P, τότε μπορούμε να την αγνοήσουμε για όλη την υπόλοιπη αναζήτηση.

Ο αλγόριθμος αναζήτησης εγγύτερου γείτονα στο ball-tree εξετάζει τους κόμβους σε σειρά ανάλογα με το βάθος, ξεκινώντας από τη ρίζα. Κατά τη διάρκεια της αναζήτησης, ο αλγόριθμος διατηρεί μια ουρά προτεραιότητας max-first (συχνά υλοποιείται σε μια σωρό) από τα εγγύτερα αντικείμενα k που συναντήθηκαν μέχρι στιγμής , που υποδηλώνεται με Q εδώ. Σε κάθε κόμβο B, μπορεί να εκτελέσει μία από τις τρεις λειτουργίες, πριν επιστρέψει τελικά μια ενημερωμένη έκδοση της ουράς προτεραιότητας:

- Εάν η απόσταση από το αντικείμενο δοκιμής T στον τρέχοντα κόμβο B είναι μεγαλύτερη από το πιο απομακρυσμένο αντικείμενο στο Q, τοτε αγνοεί το B και επιστρέφει το Q.

- Εάν το B είναι κόμβος φύλλων, σαρώνει κάθε αντικείμενο που απαριθμείται μεσα στο B και ενημερώνει κατάλληλα την ουρά εγγυτέρων γειτόνων. Και επιστρέφει την ενημερωμένη ουρά.

- Εάν το B είναι ένας εσωτερικός κόμβος, καλεί τον αλγόριθμο αναδρομικά στα δύο παιδιά του B, αναζητώντας πρώτα το παιδί του οποίου το κέντρο είναι πιο κοντά στο T. Μέτα επιστέφει την ουρά αφού κάθε μία από αυτές τις κλήσεις την έχει ενημερώσει με τη σειρά της.

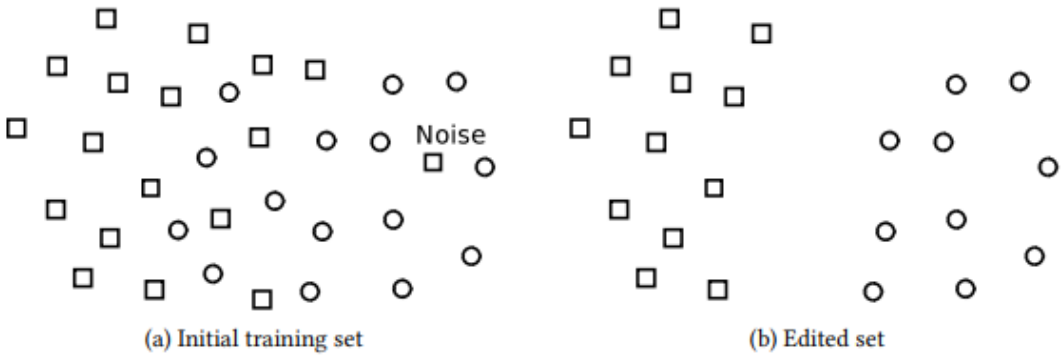
Η εκτέλεση της αναδρομικής αναζήτησης με τη σειρά που περιγράφεται στο τελευταίο βήμα παραπάνω αυξάνει την πιθανότητα ότι το πιο απομακρυσμένο παιδί θα αγνοηθεί εξ ολοκλήρου κατά τη διάρκεια της αναζήτησης.

2.3 Τεχνικές Μείωσης Δεδομένων

Πέρα από τις μεθόδους δεικτοδότησης ένας άλλος τρόπος για να αυξήσουμε την ταχύτητα του κατηγοριοποιητή k-NN είναι οι τεχνικές μείωσης δεδομένων (Data Reduction Techniques, DRTs, ΤΜΔ). Υπάρχουν δύο κατηγορίες τεχνικών μείωσης δεδομένων: οι αλγόριθμοι επιλογής προτύπου (prototype selection) [23] και αφαίρεσης προτύπων (prototype abstraction) [40]. Επιπλέον, οι αλγόριθμοι επιλογής προτύπων διακρίνονται σε αλγόριθμους συμπύκνωσης και αλγορίθμους επεξεργασίας. Οι αλγόριθμοι συμπύκνωσης και οι αλγόριθμοι αφαίρεσης προτύπων μπορούν να αντιμετωπίσουν τις αδυναμίες του κατηγοριοποιητή k-NN όσον αφορά το υψηλό υπολογιστικό κόστος και τις απαιτήσεις αποθήκευσης. Αυτό επιτυγχάνεται με την κατασκευή ενός μικρού αντιπροσωπευτικού συνόλου δεδομένων από το αρχικό σύνολο εκπαίδευσης. Αυτό το σύνολο ονομάζεται σύνολο συμπύκνωσης και περιέχει μόνο τα βασικά αντικείμενα που είναι απαραίτητα για την κατηγοριοποίηση. Εφαρμόζοντας τον κατηγοριοποιητή k-NN χρησιμοποιώντας το σύνολο συμπύκνωσης, υπάρχει πολύ χαμηλότερο υπολογιστικό κόστος και απαιτήσεις αποθήκευσης, ενώ η ακρίβεια παραμένει υψηλή ή δεν υποβαθμίζεται σημαντικά σε σχέση με την κατηγοριοποίηση στο αρχικό σύνολο εκπαίδευσης.

Από την άλλη, οι αλγόριθμοι επεξεργασίας [41] βελτιώνουν την ποιότητα των δεδομένων εκπαίδευσης με την αφαίρεση των ακραίων τιμών, του θορύβου και των λανθασμένων αντικείμενων, καθώς και με την εξομάλυνση των ορίων απόφασης της κλάσης. Στην ιδανική περίπτωση, μια διαδικασία επεξεργασίας προσπαθεί να δημιουργήσει ένα σύνολο εκπαίδευσης χωρίς επικαλύψεις μεταξύ των κλάσεων. Το σχήμα 2.4 παρουσιάζει τα δεδομένα που προσπαθούν να αφαιρέσουν οι αλγόριθμοι επεξεργασίας. Σε αυτή την ενότητα αναλύεται λεπτομερώς ο αλγόριθμος ENN-rule που είναι ο πρώτος και πιο γνωστός αλγόριθμος αυτού του είδους.

Οι ΤΜΔ μπορούν να αξιολογηθούν χρησιμοποιώντας τρία κριτήρια. Το πρώτο είναι το ποσοστό μείωσης, το οποίο δείχνει πόσο μικρότερο είναι το μέγεθος του συνόλου συμπύκνωσης σε σχέση με το μέγεθος του αρχικού συνόλου εκπαίδευση. Πρακτικά, είναι ο λόγος του αριθμού των απορριφθέντων αντικειμένων πάνω από τον αριθμό των αρχικών αντικειμένων του συνόλου



Σχήμα 2.4: Εξομαλύνοντας τα όρια των κλάσεων και αφαιρώντας τον θόρυβο [1]

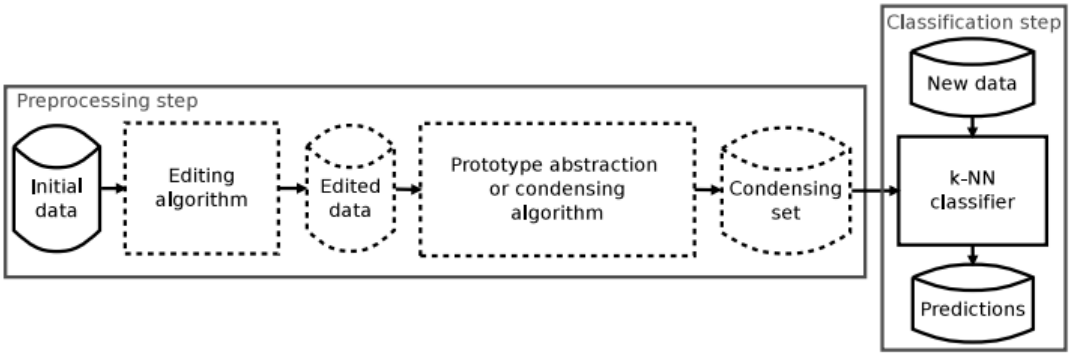
εκπαίδευσης. Προφανώς, όσο υψηλότερο είναι το ποσοστό μείωσης, τόσο ταχύτερη είναι η κατηγοριοποίηση με τον αλγόριθμο k-NN. Ένα άλλο σημαντικό κριτήριο είναι η ακρίβεια της κατηγοριοποίησης που επιτυγχάνεται από τον κατηγοριοποιητή k-NN όταν τρέχει πάνω από το σύνολο συμπύκνωσης. Το τρίτο κριτήριο είναι το υπολογιστικό κόστος προ-επεξεργασίας, το οποίο είναι, το κόστος που απαιτείται για την κατασκευή του συνόλου συμπύκνωσης. Για ορισμένους τομείς, ένα κριτήριο μπορεί να είναι πιο κρίσιμο από ένα άλλο. Ωστόσο, όλα αυτά πρέπει να πληρούν ορισμένες ελάχιστες απαιτήσεις.

Παρόλο που οι αλγόριθμοι επεξεργασίας έχουν εντελώς διαφορετικό στόχο από τις υπόλοιπες ΤΜΔ, μπορούμε να τους χρησιμοποιήσουμε για να βελτιώσουμε την απόδοσή των υπολοίπων ΤΜΔ αυξάνοντας τα ποσοστά μείωσης ή και τα επίπεδα ακρίβειας. Πιο συγκεκριμένα, τα ποσοστά μείωσης πολλών αλγορίθμων αφαίρεσης πρωτότυπων και αλγορίθμων συμπύκνωσης εξαρτώνται από το επίπεδα του θορύβου στα δεδομένα εκπαίδευσης. Τα υψηλά επίπεδα θορύβου στο σύνολο εκπαίδευσης εμποδίζουν πολλούς αλγόριθμους συμπύκνωσης ή αφαίρεσης προτύπων να επιτύχουν υψηλά ποσοστά μείωσης. Στην πραγματικότητα, όσο υψηλότερο είναι το επίπεδο θορύβου, τόσο χαμηλότερα είναι τα ποσοστά μείωσης που επιτυγχάνονται. επομένως, η αποτελεσματική εφαρμογή τέτοιων αλγορίθμων συνεπάγεται με την αφαίρεση του θορύβου από τα δεδομένα [42]. Ως εκ τούτου, ένας αλγόριθμος επεξεργασίας θα πρέπει να χρησιμοποιείται σε ένα σύνολο εκπαίδευσης με θόρυβο, προκειμένου είτε να βελτιωθεί η ακρίβεια είτε να γίνει πιο αποτελεσματική η εφαρμογή αλγορίθμων συμπύκνωσης και αφαίρεσης προτύπων.

Το σχήμα 2.5 συνοψίζει τη διαδικασία κατηγοριοποίησης του αλγορίθμου k-NN μέσω της μείωσης των δεδομένων. Η όλη διαδικασία περιλαμβάνει δύο φάσεις, προ-επεξεργασία και κατηγοριοποίησης . Η φάση προ-επεξεργασίας είναι προαιρετική. Γενικά, υπάρχουν τέσσερις πιθανοί τύποι προ-επεξεργασίας:

1. χωρίς προ-επεξεργασία

2. μόνο χρήση αλγόριθμου επεξεργασία



Σχήμα 2.5: Κατηγοριοποίηση εγγυτέρων γειτόνων μέσω μείωσης δεδομένων [1]

3. μόνο χρήση αλγορίθμου συμπύκνωσης
4. χρήση και αλγορίθμου επεξεργασίας και μετά αλγορίθμου συμπύκνωσης

Εάν το σύνολο εκπαίδευσης δεν περιέχει θόρυβο και παραπλανητικά δεδομένα άλλα και το μέγεθός του είναι μικρό, τότε δεν απαιτείται καθόλου προ-επεξεργασία. Όταν το μέγεθος του συνόλου εκπαίδευσης είναι μικρό, αλλά περιέχει θόρυβο, χρειάζεται να εκτελεστεί μόνο ένας αλγόριθμος επεξεργασίας κατά την προ-επεξεργασία. Από την άλλη πλευρά, σε περιπτώσεις μεγάλων συνόλων εκπαίδευσης τα οποία όμως δεν περιέχουν αντικείμενα που θεωρούνται θόρυβος, θα πρέπει να εκτελείται μόνο ένας αλγόριθμος μείωση δεδομένων χωρίς να έχει προηγηθεί κάποιος αλγόριθμος επεξεργασία στο σύνολο εκπαίδευσης. Τέλος, σε περιπτώσεις μεγάλων συνόλων εκπαίδευσης που όμως περιέχουν και θόρυβο, πρέπει να εκτελούνται και τα δύο είδη αλγορίθμων προ-επεξεργασίας. Σε αυτήν την πτυχιακή εργασία θα εκτελέσουμε στο κάθε σύνολο δεδομένων και τους τέσσερις πιθανούς τύπους προ-επεξεργασίας.

Ο στόχος μιας διαδικασίας πλήρους προ-επεξεργασίας μείωσης δεδομένων είναι η κατασκευή ενός συνόλου συμπύκνωσης χωρίς θόρυβο διατηρώντας ή δημιουργώντας για κάθε κλάση επαρκή αριθμό προτύπων που είναι απαραίτητα για την κατηγοριοποίηση k εγγυτέρων γειτόνων. Θα πρέπει να αναφέρουμε ότι πολλές ΤΜΔ έχουν εφαρμοστεί στο πλαίσιο του λογισμικού KEEL [43], από το οποίο περνούμε τα περισσότερα σύνολα δεδομένων αυτής της πτυχιακής εργασίας, το οποίο είναι ένα πλαίσιο ανοιχτού κώδικα που βασίζεται σε Java. Επίσης υπάρχουν υλοποιημένες ορισμένες ΤΜΔ στη βιβλιοθήκη imbalanced-learn της Python [4], οι οποίες είναι κυρίως για σύνολα δεδομένων με άνισες κλάσεις, όμως με τις κατάλληλες παραμέτρους μπορούν να εφαρμοστούν σε οποιοδήποτε σύνολο δεδομένων.

2.3.1 ENN-rule

Ο αλγόριθμος επεξεργασίας εγγύτερου γείτονα (Edited Nearest Neighbor rule, ENN-rule), εφευρέθηκε από τον Dennis Wilson [41] και είναι ο πρώτος και ένας από τους πιο γνωστούς αλγορίθμους επιλογής προτύπου για την επεξεργασία δεδομένων. Ο ENN-rule αποτελεί τη βάση όλων των άλλων αλγορίθμων επεξεργασίας.

Αρχικά, το επεξεργασμένο σύνολο (ES) έχει οριστεί να είναι ίσο με το σύνολο εκπαίδευσης (TS). Μετά για κάθε αντικείμενο x του TS, ο αλγόριθμος σαρώνει το TS και ανακτά τους k εγγυτέρους γείτονές του. Συνεχίζοντας εάν το x έχει κατηγοριοποιηθεί εσφαλμένα από την πλειοψηφία των εγγύτερων γειτόνων του τότε αφαιρείται από το ES. Ο ENN-rule θεωρεί ότι τα εσφαλμένα κατηγοριοποιημένα αντικείμενα είναι θόρυβος ή αντικείμενα κοντά στα σύνορα και, ως εκ τούτου, πρέπει να αφαιρεθούν. Σε κάθε επανάληψη του αλγορίθμου, ο ENN-rule κάνει αναζητήσεις για τους εγγυτέρους γείτονες στο αρχικό σύνολο εκπαίδευσης και όχι στο υπό κατασκευή επεξεργασμένο σύνολο. Ο αλγόριθμος περιγράφεται σε ψευδογλώσσα παρακάτω.

Όπως είναι εμφανές, το κόστος της επεξεργασίας εξαρτάται από το μέγεθος του συνόλου εκ-

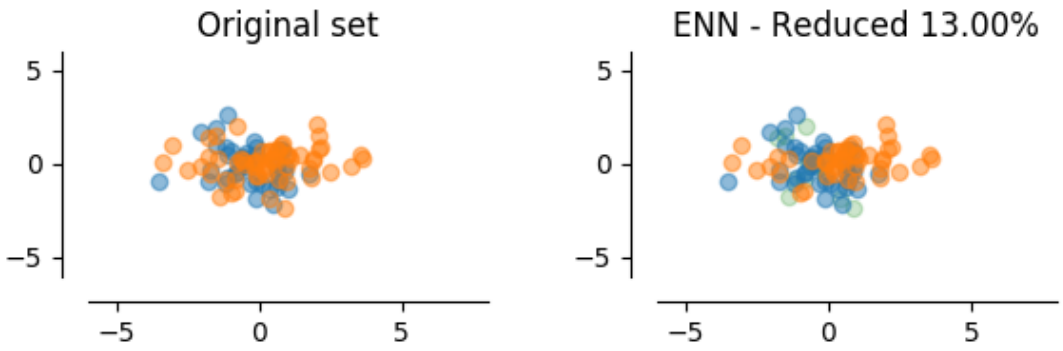
παίδευσης. Σε περιπτώσεις μεγάλων συνόλων δεδομένων, η εκτέλεση του αλγορίθμου του ENN-rule είναι μια χρονοβόρα διαδικασία. Θεωρώντας ως N των αριθμό των αντικειμένων στο σύνολο εκπαίδευσης, ο αλγόριθμος θα πρέπει να υπολογίσει $\frac{N \times (N-1)}{2}$ αποστάσεις.

Ψευδοκώδικας ENN-rule [1]

Input: TS, k
Output: ES
1: $ES \leftarrow TS$
2: **for** each $x \in TS$ **do**
3: $NNs \leftarrow$ find the k nearest to x neighbors in $TS - \{x\}$
4: $majorClass \leftarrow$ find the most common class of NNs
5: **if** $x_{class} \neq majorClass$ **then**
6: $ES \leftarrow ES - \{x\}$
7: **end if**
8: **end for**
9: **return** ES

Ένα κρίσιμο ζήτημα που χρειαστεί να αντιμετωπιστεί είναι ο προσδιορισμός της τιμής που θα αποδώσουμε στο k που καθορίζει το μέγεθος της εξεταζόμενης γειτονιάς. Συχνά θεωρούμε το k = 3 ως την τυπική τιμή [44]. Αυτό υιοθετείται σε πολλές έρευνες, ενώ άλλες έρευνες χρησιμοποιούν k = 3 και πρόσθετες τιμές k [45]. Σε ορισμένες περιπτώσεις, οι ερευνητές καθορίζουν την αξία του k που επιτυγχάνει την καλύτερη απόδοση μέσω διαδικασιών δοκιμής και σφάλματος [46]. Επιπλέον, στο [47], γίνεται μια πειραματική αξιολόγηση ενός μεγάλου αριθμού τιμών K. Αυτή η έρευνα αποδεικνύει ότι η καλύτερη τιμή του k δεν είναι μια συγκεκριμένη τιμή άλλα εξαρτάται από το κάθε σύνολο δεδομένων. Και θα πρέπει να καθορίζεται με με βάση την κατανομή των αντικειμένων του κάθε συνόλου στον πολυδιάστατο χώρο.

Ακόμη και η καλύτερη τιμή του k μπορεί να μην είναι η βέλτιστη και μπορεί να αφαιρέσει α-



Σχήμα 2.6: Σύνολο δεδομένων πριν και μετά την χρήση του ENN [4]

ντικείμενα που δεν είναι θόρυβος [48] ή να κρατήσει τα αντικείμενα που είναι όντως θόρυβος. Αυτό συμβαίνει επειδή ο ENN-rule χρησιμοποιεί μια μοναδική τιμή k για ολόκληρο το σύνολο εκπαίδευσης. Ενώ μια τιμή k μπορεί να είναι καλή σε μια μόνο περιοχή του συνόλου εκπαίδευσης, δηλαδή διαφορετικές τιμές k μπορεί να είναι βέλτιστες για διαφορετικές περιοχές στο χώρο.

2.3.2 CNN-rule

Ο αλγόριθμος συμπύκνωσης εγγύτερου γείτονα (Condensing Nearest Neighbour rule, CNN-rule) [49] είναι μια ΤΜΔ και είναι ο πρώτος και ένας αλγόριθμος αναφοράς για συμπύκνωση δεδομένων. Χρησιμοποιείται σε πολλά άρθρα για λόγους σύγκρισης. Ο CNN-rule είναι μια από τις ΤΜΔ που χτίζει το σύνολο συμπύκνωσης έχοντας ως βάση μια απλή ιδέα.

Η ιδέα αυτή είναι ότι τα αντικείμενα που βρίσκονται στην εσωτερική περιοχή δεδομένων μιας κλάσης (δηλαδή μακριά από τα σύνορα της κάθε κλάσης) δεν είναι σημαντικά κατά τη διαδικασία κατηγοριοποίηση. Κατά συνέπεια, μπορούν να διαγράφουν χωρίς σημαντική απώλεια ακρίβειας. Υιοθετώντας αυτή την ιδέα, ο CNN-rule προσπαθεί να τοποθετήσει στο σύνολο συμπύκνωσης μόνο τα αντικείμενα που βρίσκονται κοντά στα όρια μεταξύ κλάσεων. Γιατί αυτά είναι τα μόνα απαραίτητα αντικείμενα για την διαδικασία της κατηγοριοποίησης. Το σχήμα 2.8 απεικονίζει αυτή τη στρατηγική. Η ιδέα είναι ότι ο κατηγοριοποιητής k εγγυτέρων γειτόνων θα είναι σε θέση να έχει παρόμοια ακρίβεια χρησιμοποιώντας είτε το σύνολο εκπαίδευσης είτε το σύνολο συμπύκνωσης. Ωστόσο, η χρήση του το συμπυκνωμένου σύνολου περιλαμβάνει πολύ χαμηλότερες υπολογιστικές απαιτήσεις και δαπάνες αποθήκευσης από το αρχικό σύνολο εκπαίδευσης.

Ο CNN-rule προσπαθεί να κρατήσει τα αντικείμενα τα οποία βρίσκονται κοντά στα σύνορά μεταξύ των κλάσεων ως εξής. Αρχικά, ένα αντικείμενο του συνόλου εκπαίδευσης (TS) μετακινείται στο σύνολο συμπύκνωσης (CS). Μέτα, ο CNN-rule εφαρμόζει τον κανόνα του ενός εγγύτερου γείτονα και κατηγοριοποιεί τα αντικείμενα του συνόλου εκπαίδευσης σαρώνοντας τα αντικείμενα του σύνολου συμπύκνωσης. Εάν ένα αντικείμενο έχει κατηγοριοποιηθεί εσφαλμένα, μετακινείται από το σύνολο εκπαίδευσης στο σύνολο συμπύκνωσης. Ο αλγόριθμος συνεχίζει μέχρι να μην υπάρχουν άλλες μετακινήσεις από το συνόλου εκπαίδευσης στο σύνολο συμπύκνωσης κατά τη διάρκεια μιας πλήρους σάρωσης του συνόλου εκπαίδευσης. Αυτό μας διασφαλίζει ότι το περιεχόμενο του συνόλου εκπαίδευσης κατηγοριοποιείται σωστά από το περιεχόμενο του σύνολο συμπύκνωσης. Το υπόλοιπο περιεχόμενο του συνόλου εκπαίδευσής διαγράφεται. Αυτή η διαδικασία περιγράφεται στον παρακάτω ψευδωκώδικα.

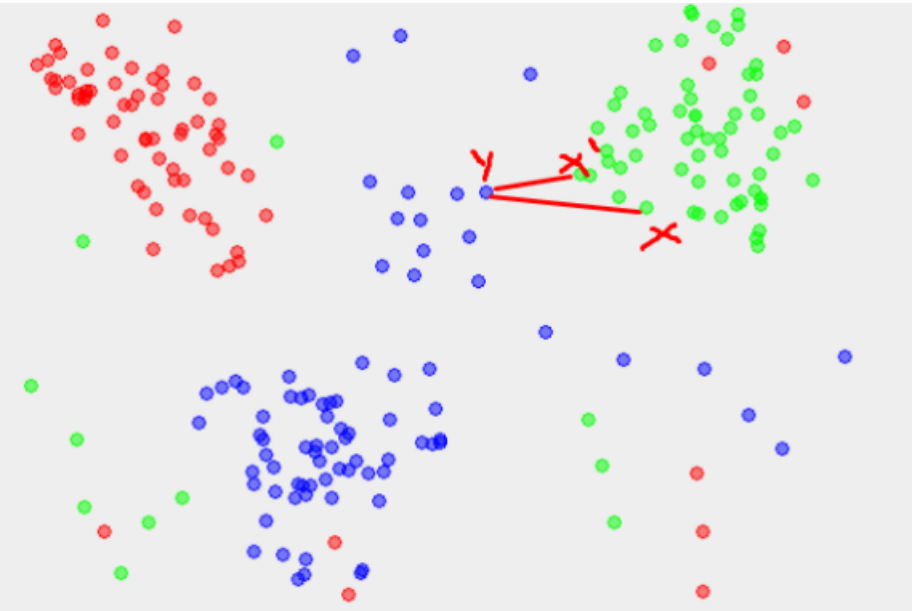
Ένα πλεονέκτημα του CNN-rule είναι ότι παίρνει μια μη παραμετρική προσέγγιση. Καθορίζει τον αριθμό των προτύπων αυτόματα, χωρίς να χρειάζεται ο χρήστης να καθορίσει κάποια παράμετρο πριν από την εκτέλεση του αλγορίθμου. Μια άλλη επιθυμητή ιδιότητα είναι ότι ο κανόνας μέσω των πολλαπλών περασμάτων πάνω από τα δεδομένα εγγυάται πως τα αφαι-

ρούμενα αντικείμενα εκπαίδευσης μπορούν να κατηγοριοποιηθούν σωστά στα πλαίσια του τελικού συμπυκνωμένου συνόλου που δημιουργήθηκε, δηλαδή αν εκτελέσουμε τον αλγόριθμο 1-εγγύτερου γείτονα στα αντικείμενα εκπαίδευσης που δεν κατάφεραν να μπουν στο νέο αυτό συμπυκνωμένο σύνολο.

Ψευδοκώδικας CNN-rule [1]
Input: TS
Output: CS
1: $CS \leftarrow \varnothing$
2: pick an item of TS and move it to CS
3: **repeat**
4: $stop \leftarrow TRUE$
5: **for** each $x \in TS$ **do**
6: $NN \leftarrow$ Nearest Neighbour of x in CS
7: **if** $NN_{class} \neq x_{class}$ **then**
8: $CS \leftarrow CS \cup \{x\}$
9: $TS \leftarrow TS - \{x\}$
10: $stop \leftarrow FALSE$
11: **end if**
12: **end for**
13: **until** $stop == TRUE$ {no move during a pass of TS}
14: **discard** TS
15: **return** CS

Ο CNN-rule θεωρεί ότι τα λάθος κατηγοριοποιημένα αντικείμενα είναι πιθανώς κοντά στα όρια αποφάσεων των κλάσεων και έτσι πρέπει να τοποθετηθούν στο σύνολο συμπύκνωσης. Προφανώς, δημιουργείται ένα άλλο πρόβλημα με αυτόν τον τρόπο. Τα αντικείμενα που είναι θόρυβος, δηλαδή αυτά τα αντικείμενα που ανήκουν σε μια διαφορετική κλάση από τα αντικείμενα τριγύρω τους, μας παραπλανούν. Ο CNN-rule τοποθετεί λανθασμένα αυτό το είδος των αντικειμένων άλλα και όσα αντικείμενα βρίσκονται κοντά τους στο σύνολο συμπύκνωσης, γιατί θεωρεί ότι είναι στα των όρια κλάσεων. Για το λόγο αυτό, το ποσοστό μείωσης και το κόστος προ-επεξεργασίας επηρεάζονται από υψηλά επίπεδα θορύβου.

Φυσικά, ο αριθμός των διακριτών κλάσεων μπορεί επίσης να επηρεάσει το ποσοστό μείωσης. Όσες περισσότερες κλάσεις υπάρχουν σε ένα σύνολο δεδομένων, τόσα περισσότερα όρια πιθανόν να υπάρχουν και ως εκ τούτου συλλέγονται περισσότερα πρωτότυπα για να τοποθετήσουμε στο σύνολο συμπύκνωσης. Ένα ακόμα μειονέκτημα είναι ότι το σύνολο συμπύκνωσης που προκύπτει εξαρτάται από την σειρά των αντικειμένων του συνόλου εκπαίδευσής. Ως εκ τούτου, ο CNN-rule χτίζει ένα διαφορετικό σύνολο συμπύκνωσης εξετάζοντας ακριβώς τα ίδια δεδομένα εκπαίδευσης αλλά με διαφορετική σειρά.

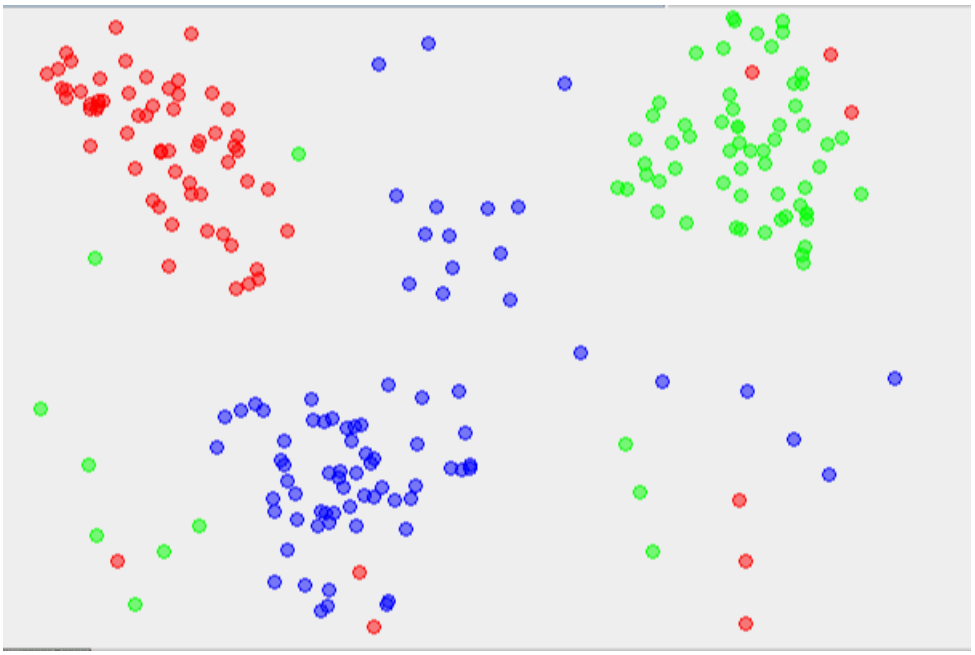


Σχήμα 2.7: αναλογία συνόρων

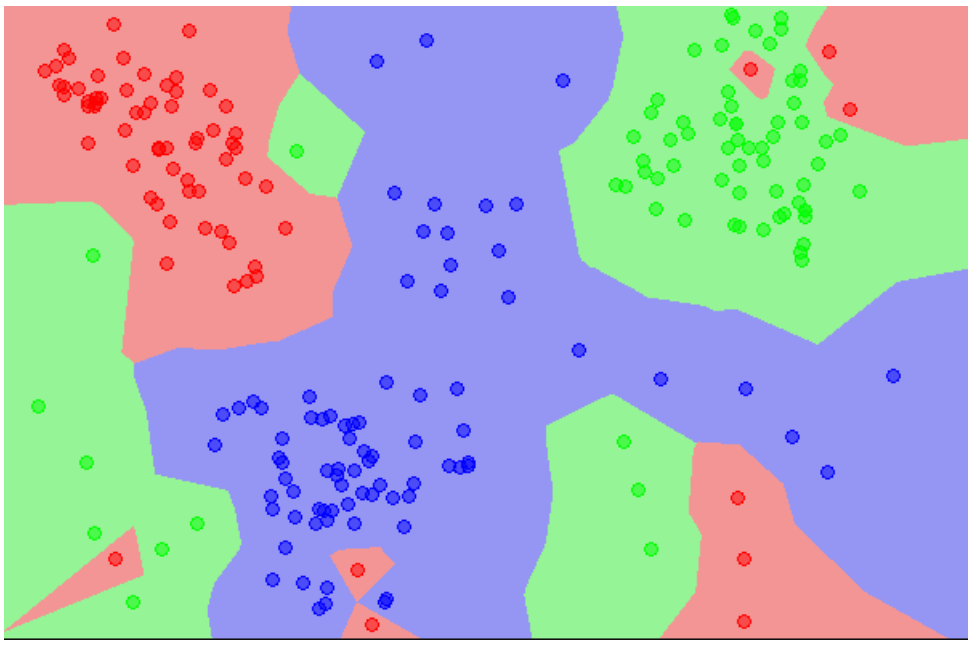
Επιπλέον, ο CNN-rule είναι non-incremental δηλαδη δεν μπορεί να διαχειριστεί νέα δεδομένα εκπαίδευσης. Αυτό σημαίνει ότι τα νέα δεδομένα εκπαίδευσης δεν μπορούν να ενημερώσουν ένα ήδη κατασκευασμένο σύνολο συμπύκνωσης. Ο CNN-rule κάνει πολλαπλά περάσματα πάνω από τα δεδομένα και δεν μπορεί να χρησιμοποιήσει τα μετέπειτα διαθέσιμα δεδομένα εκπαίδευσης για να ενημερώσει το σύνολο συμπύκνωσης. Για να κατασκευαστεί ένα ενημερωμένο σύνολο συμπύκνωσης προσθέτοντας τα καινούργια δεδομένα, θα πρέπει να εκτελείται από το μηδέν πάνω από το πλήρες σύνολο εκπαίδευσης (νέα και παλιά δεδομένα εκπαίδευσης). Άρα, τα αντικείμενα εκπαίδευσης που δεν εισέρχονται στο σύνολο συμπύκνωσης θα έπρεπε να διατηρούνται. Ως εκ τούτου, CNN-rule είναι ακατάλληλος για δυναμικά ή streaming περιβάλλοντα [50] όπου τα νέα δεδομένα εκπαίδευσης είναι σταδιακά διαθέσιμα. Επιπλέον, ο CNN-rule απαιτεί όλα τα αντικείμενα εκπαίδευσης να είναι κάτοικοι μνήμης.

Είναι αποτελεσματικό να σαρώσουμε τα αντικείμενα εκπαίδευσης κατά φθίνουσα σειρά αναλογίας συνόρων (border ratio) [51, 52]. Η αναλογία συνόρων ενός αντικείμενου εκπαίδευσης x ορίζεται ως $a(x) = \frac{\| x' - y \|}{\| x - y \|}$ οπού το $\| x - y \|$ είναι η απόσταση από το πλησιέστερο παράδειγμα y που ανήκει σε διαφορετική κλάση από το x, και το $\| x' - y \|$ είναι η απόσταση από το y μέχρι το πλησιέστερο παράδειγμα x' με την ίδια ετικέτα με το x.

Ο λόγος συνόρων είναι στο διάστημα $[0, 1]$ επειδή $\| x' - y \|$ποτέ δεν υπερβαίνει $\| x - y \|$. Αυτή η ταξινόμηση δίνει το προνόμιο στα σύνορα των κλάσεων να συμπεριληφθούν στο σύνολο των προτύπων s. Ένα αντικείμενο διαφορετικής κλάσης από το x ονομάζεται εξωτερικό προς το x. Ο υπολογισμός του λόγου των συνόρων απεικονίζεται στο σχήμα 2.7. Τα αντικείμενα δεδομένων επισημαίνονται με χρώματα: το αρχικό αντικείμενο είναι το x και η ετικέτα του είναι πράσινο. Τα εξωτερικά αντικείμενα είναι μπλε και κόκκινα. Το πλησιέστερο στο x εξωτερικό αντικείμενο είναι το y. Το εγγύτερο στο y πράσινο αντικείμενο είναι το x'. Ο λόγος συνόρων



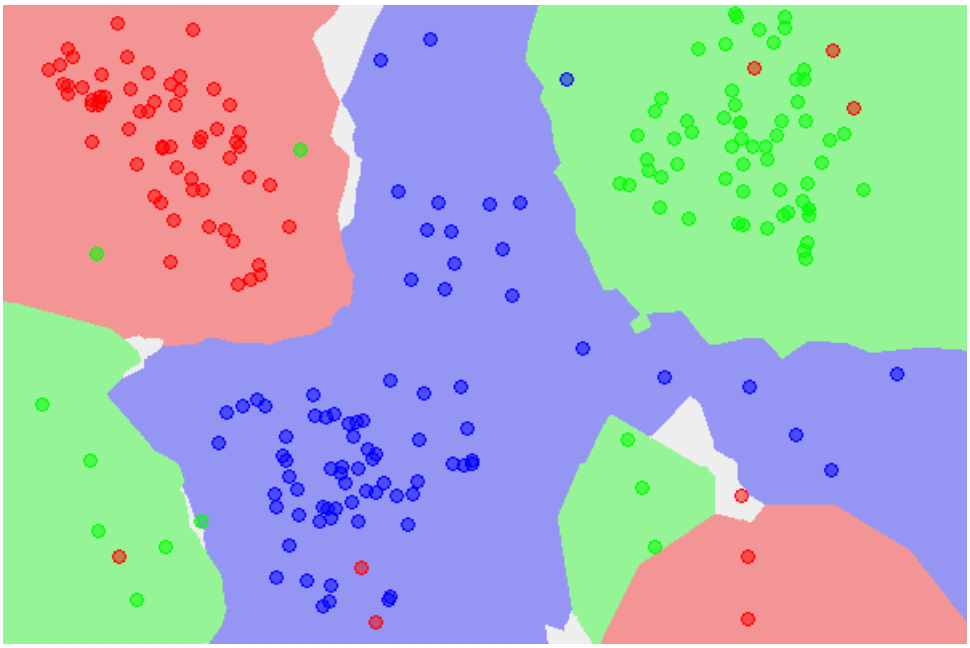
2.8.1. Το σύνολο δεδομένων



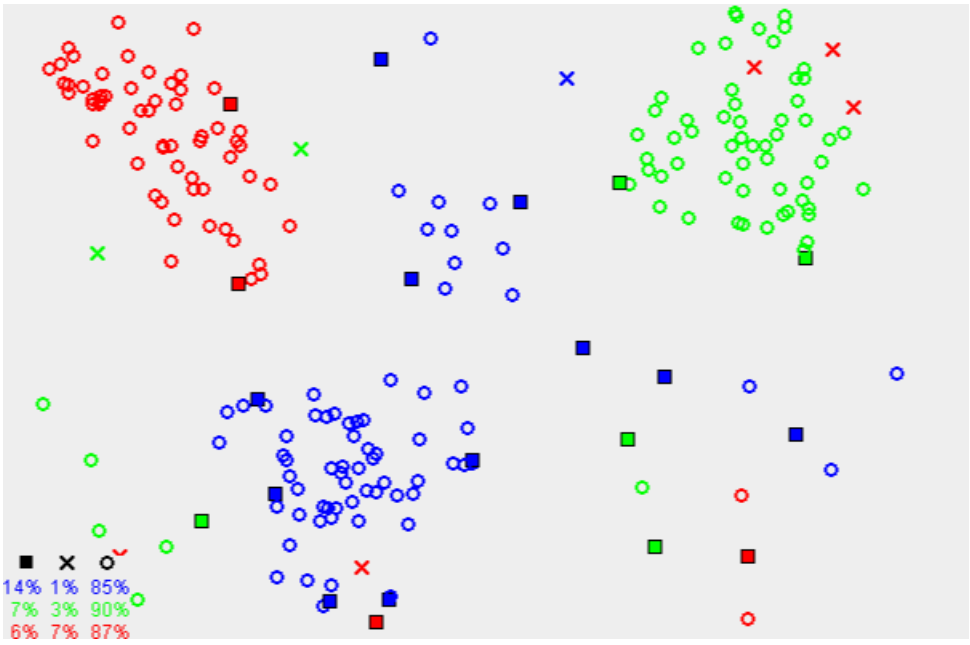
2.8.2. Ο χάρτης 1-NN

$a(x)$ είναι το χαρακτηριστικό του αρχικού αντικειμένου x.

Στο σχήμα 2.8 φαίνεται η διαδικασία του CNN-rule. Στο σχήμα 2.8.1 έχουμε ένα σύνολο δεδομένων με 3 κλάσεις ανάλογά με το χρώμα (κόκκινο, μπλε, πράσινο). Στο σχήμα 2.8.2 έχουμε το χάρτη του ενός εγγύτερου γείτονα, κάθε αντικείμενο κατηγοριοποιείται ανάλογα με τον πιο κοντινό του γείτονα. Στο σχήμα 2.8.3 έχουμε τον χάρτη των 5-εγγυτέρων γειτόνων. Οι λευκές περιοχές αντιστοιχούν στις μη κατηγοριοποιημένες περιοχές, όπου η ψηφοφορία 5-NN έχει λήξει σε ισοπαλία (για παράδειγμα, εάν υπάρχουν ένα πράσινο, δύο κόκκινα και δυο μπλε αντικείμενα μεταξύ 5 εγγύτερων γειτόνων). Στο σχήμα 2.8.4 έχουμε το μειωμένο σύνολο δεδομένων

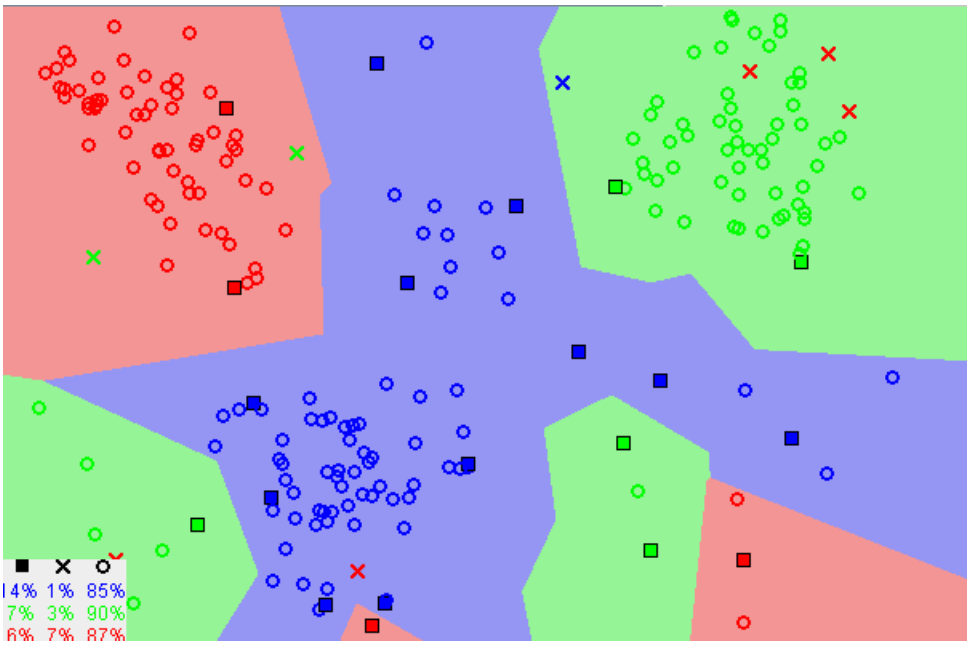


2.8.3. Ο χάρτης 5-NN



2.8.4. Εφαρμογή του CNN-rule

από τον CNN-rule. Οι σταυροί είναι οι ακραίες τιμές των κλάσεων που επιλέγονται από τον κανόνα (3,2)-NN δηλαδή και οι τρεις εγγύτεροι γείτονες αυτών των αντικείμενα ανήκουν σε άλλες κλάσεις. Τα τετράγωνα είναι τα πρωτότυπα και οι άδειοι κύκλοι είναι τα απορροφημένα αντικείμενα. Η κάτω αριστερή γωνία δείχνει τους αριθμούς της κάθε κλάσης για ακραίες τιμές, πρωτότυπα και απορροφημένα αντικείμενα. Ο αριθμός των προτύπων κυμαίνεται από 6 % έως 14 % για διαφορετικές κλάσεις σε αυτό το παράδειγμα. Στο σχήμα 2.8.5 έχουμε τον χάρτη κατηγοριοποίησης 1-NN με τα πρωτότυπα και παρατηρούμε ότι είναι παρόμοιος με αυτόν με το αρχικό σύνολο δεδομένων. Τα σχήματά παρήχθησαν χρησιμοποιώντας τη μικρο-εφαρμογή του Mirkes [51].



2.8.5. Ο χάρτης 1-NN μετά την εφαρμογή του CNN-rule

Σχήμα 2.8: κατηγοριοποίηση K-NN μέσω CNN-rule

2.3.3 IB2

Ψευδοκώδικας IB2 [1]
Input: TS
Output: CS
1:$CS \leftarrow \varnothing$
2: pick an item of TS and move it to CS
3: **for** each $x \in TS$ **do**
4: $NN \leftarrow$ Nearest Neighbour of x in CS
5: **if** $NN_{class} \neq x_{class}$ **then**
6: $CS \leftarrow CS \cup \{x\}$
7: **end if**
8: $TS \leftarrow TS - \{x\}$
9: **end for**
10: **return** CS

Ο επόμενος αλγόριθμος είναι ο IB2, ο οποίος ανήκει στη γνωστή οικογένεια αλγορίθμων μάθησης μέσω παραδειγμάτων (Instance-Based Learning, IBL) [53] και βασίζεται στον CNN-rule . Στην πραγματικότητα, ο IB2 αποτελεί μια απλή παραλλαγή του CNN-rule άπλα κάνοντας μόνο ένα πέρασμα. Κάθε αντικείμενο εκπαίδευσης x του TS κατηγοριοποιείται χρησιμοποιώντας τον κατηγοριοποιητή 1-NN στο τρέχων CS. Εάν το x κατηγοριοποιηθεί σωστά, απορρίπτεται. Διαφορετικά, το x μεταφέρεται στο CS. Ο αλγόριθμος του IB2 παρουσιάζεται σε ψευδοκώδικα

παραπάνω.

Παρόμοια με τον CNN-rule, ο IB2 είναι ένας μη παραμετρικός αλγόριθμος και το σύνολο συμπύκνωσης εξαρτάται σε μεγάλο βαθμό από τη σειρά των αντικειμένων στο σύνολο εκπαίδευσης. Σε αντίθεση με τον του CNN-rule, ο IB2 δεν εξασφαλίζει ότι όλα τα αντικείμενα που απορρίφθηκαν μπορούν να κατηγοριοποιηθούν σωστά στην τελική έκδοση του συνόλου συμπύκνωσης. Ωστόσο, δεδομένου ότι είναι ένας αλγόριθμος που πραγματοποιεί ένα περάσμα, είναι πολύ γρήγορος, δηλαδή, συνεπάγεται με χαμηλό υπολογιστικό κόστος προ-επεξεργασίας.

Επιπλέον, το IB2 χτίζει το σύνολο συμπύκνωσης σταδιακά. Νέα αντικείμενα μπορούν να ληφθούν υπόψη για τη δημιουργία του σύνολου συμπύκνωσης. Με άλλα λόγια, τα νέα τμήματα δεδομένων εκπαίδευσης μπορούν να ενημερώσουν ένα υπάρχον σύνολο συμπύκνωσης με απλό τρόπο και χωρίς να ληφθούν υπόψη τα "παλιά" αφαιρούμενα δεδομένα που είχαν χρησιμοποιηθεί για την κατασκευή του συνόλου συμπύκνωσης. Κάθε νέο αντικείμενο εκπαίδευσης μπορεί να εξεταστεί, να τοποθετηθεί ή όχι στο σύνολο συμπύκνωσης και στη συνέχεια να αφαιρεθεί.

Ο IB2 μπορεί να χειριστεί νέες ετικέτες κλάσης, ως εκ τούτου, ο IB2 είναι μια κατάλληλη ΤΜΔ για δυναμικά ή και συνεχούς ροής περιβάλλοντα, όπου τα νέα δεδομένα εκπαίδευσης μπορεί να είναι σταδιακά διαθέσιμα. Τέλος, σε αντίθεση με τον CNN-rule και με πολλές άλλες ΤΜΔ, ο IB2 δεν απαιτεί ότι όλα τα δεδομένα εκπαίδευσης θα πρέπει βρίσκονται στην κύρια μνήμη. Έτσι μπορεί να εφαρμοστεί σε συσκευές των οποίων η μνήμη είναι ανεπαρκής για την αποθήκευση όλων των δεδομένων εκπαίδευσης.

2.3.4 RSP3

Ο αλγόριθμος RSP3 ειναι ο τελευταίος από τους τρεις αλγορίθμους Μείωσης μεσο Διαχωρισμού του Χώρου (Reduction by Space Partitioning, RSP) [40], οι οποίοι είναι βασισμένοι στον αλγόριθμο CJA (Chen and Jozwik Algorithm) [54] , στον οποίο και θα αναφερθούμε πρώτα ώστε να γίνει κατανοητός ο RSP3.

Ο Chen και ο Jozwik έχουν προτείνει έναν πολύ γνωστό και αποτελεσματικό αλγόριθμο αφαίρεσης προτύπων. Ο αλγόριθμος CJA βασίζεται σε μια επαναληπτική διαδικασία διαχωρισμού του αρχικού συνόλου. Αρχικά ανακτά τα πιο απομακρυσμένα αντικείμενα, x και y στο σύνολο εκπαίδευσης. Η απόσταση ε μεταξύ x και y καθορίζει τη διάμετρο του συνόλου δεδομένων. Στη συνέχεια, με βάση αυτά τα δύο αντικείμενα, ο CJA χωρίζει το σύνολο εκπαίδευσης σε δύο υποσύνολα, τα αντικείμενα που βρίσκονται πιο κοντά στο x τοποθετούνται σε Sx ενώ τα αντικείμενα που βρίσκονται πιο κοντά στο y τοποθετούνται σε Sy. Ο CJA συνεχίζει επιλέγοντας να διαιρέσει υποσύνολα που περιέχουν αντικείμενα που ανήκουν σε περισσότερες από μια κλάση (μη ομοιογενή υποσύνολα). Το μη ομοιογενές υποσύνολο με τη μεγαλύτερη διάμετρο διαιρείται πρώτο. Εάν όλα τα υποσύνολα είναι ομοιογενή, ο CJA συνεχίζει διαιρώντας τα ομοιογενή υποσύνολα, ξεκινώντας από το μεγαλύτερο. Αυτή η διαδικασία συνεχίζεται έως ότου ο αριθμός των υποσυνόλων γίνει ίσος με μια τιμή που έχει προκαθοριστεί από το χρήστη. Στο τέλος, για κάθε

υποσύνολο S που έχει δημιουργηθεί, το CJA υπολογίζει το μέσο όρο από τα αντικείμενα στο S και δημιουργεί ένα μέσο αντικείμενο στο οποίο αποδίδεται η ετικέτα της κλάσης πλειοψηφίας στο S. Τα μέσα αντικείμενα που δημιουργούνται αποτελούν το τελικό σύνολο συμπύκνωσης.

Ο CJA επιλέγει το επόμενο υποσύνολο που θα διαιρεθεί εξετάζοντας τη διάμετρο του. Η βασική ιδέα είναι ότι, ένα υποσύνολο που έχει μεγάλη διάμετρο πιθανώς θα περιέχει περισσότερα αντικείμενα εκπαίδευσης. Επομένως, εάν αυτό το υποσύνολο διαιρεθεί πρώτα, θα επιτευχθεί υψηλότερο ποσοστό μείωσης. Μια επιθυμητή ιδιότητα είναι ότι ο CJA χτίζει το ίδιο υποσύνολο συμπύκνωσης ανεξάρτητα από την σειρά των δεδομένων στο σύνολο εκπαίδευσης. Ωστόσο, έχει δύο αδύνατα σημεία. Το πρώτο είναι ότι ο αλγόριθμος είναι παραμετρικός, ο χρήστης θα πρέπει να καθορίσει τον αριθμό των προτύπων πριν από την εκτέλεση του αλγορίθμου. Και αυτό συνήθως περιλαμβάνει τον συντονισμό μέσω μιας δαπανηρής διαδικασίας δοκιμής και σφάλματος. Σε ορισμένους τομείς, αυτή η ιδιότητα μπορεί να είναι επιθυμητή, καθώς επιτρέπει σε κάποιον να ελέγχει το μέγεθος του υποσυνόλου συμπύκνωσης. Ωστόσο, απαγορεύει τον αυτόματο προσδιορισμό του μεγέθους του υποσυνόλου συμπύκνωσης σύμφωνα με τη φύση των διαθέσιμων δεδομένων.

Όπως είπαμε και πιο πριν ο αλγόριθμος των Chen και Jozwik αποτελεί τον πρόγονο της οικογένειας των αλγορίθμων μείωσης μέσο διαχωρισμό χώρου (RSP) που είναι ένα δημοφιλές σύνολο τριών πρωτότυπων αλγορίθμων αφαίρεσης γνωστών ως RSP1, RSP2 και RSP3. Ο RSP1 προσπαθεί να διορθώσει το δεύτερο μειονέκτημα του CJA. Πιο συγκεκριμένα, ο RSP1 υπολογίζει τόσα μέσα αντικείμενα όσα και ο αριθμός των διαφορετικών κλάσεων μέσα σε κάθε υποσύνολο. Ως εκ τούτου, υπολογίζει το μέσο όρο των αντικειμένων που ανήκουν σε κάθε κλάση σε ένα υποσύνολο. Προφανώς, το RSP1 χτίζει μεγαλύτερα σύνολα συμπύκνωσης από το CJA. Ωστόσο, ο RSP1 προσπαθεί να βελτιώσει την ακρίβεια, δεδομένου ότι λαμβάνει υπόψη όλα τα αντικείμενα εκπαίδευσης (δηλαδή δεν αγνοεί τα αντικείμενα).

Οι αλγόριθμοι RSP1 και οι RSP2 διαφέρουν ως προς τον τρόπο με τον οποίο επιλέγουν το επόμενο υποσύνολο που θα διαχωριστεί. Παρόμοια με τον CJA, ο RSP1 χρησιμοποιεί τη διάμετρο του υποσυνόλου ως το κριτήριο του διαχωρισμού, με βάση την ιδέα ότι το υποσύνολο με τη μεγαλύτερη διάμετρο λογικά θα περιέχει περισσότερα αντικείμενα εκπαίδευσης και έτσι θα μπορούσε να επιτευχθεί υψηλότερος βαθμός μείωσης του συνόλου. Αντίθετα, ο RSP2 χρησιμοποιεί ως κριτήριο του διαχωρισμού τον υψηλότερο βαθμό επικάλυψης. Αυτό το κριτήριο προϋποθέτει ότι τα αντικείμενα που ανήκουν σε μια συγκεκριμένη κλάση βρίσκονται όσο το δυνατόν πιο κοντά το ένα στο άλλο, ενώ τα αντικείμενα που ανήκουν σε διαφορετικές κλάσεις βρίσκονται όσο το δυνατόν πιο μακριά το ένα από το άλλο. Ο καλύτερος τρόπος από τους δυο αυτούς είναι να διαιρέσουμε το υποσύνολο με τον υψηλότερο βαθμό επικάλυψης [40]. Ο βαθμός επικάλυψης ενός υποσυνόλου είναι ο λόγος της μέσης απόστασης μεταξύ αντικειμένων που ανήκουν σε διαφορετικές κλάσεις και της μέσης απόστασης μεταξύ αντικειμένων που ανήκουν στην ίδια κλάση.

Ο RSP3 υιοθετεί την έννοια της ομοιογένειας. Συνεχίζει τη διάσπαση των μη ομοιογενών υποσυνόλων και τερματίζει όταν όλα αυτά γίνουν ομοιογενή, δηλαδή ο RSP3 μπορεί να χρησιμοποιήσει είτε τη μεγαλύτερη διάμετρο είτε τον υψηλότερο βαθμό επικάλυψης ως κριτήριο του διαχωρισμού του υποσυνόλου. Στην πραγματικότητα, δεδομένου ότι όλα τα μη ομοιογενή υποσύνολα διαιρούνται, η επιλογή του κριτηρίου διαχωρισμού γίνεται ζήτημα μικρότερης σημασίας. Ο RSP3 είναι ο μόνος αλγόριθμος της οικογένειας RSP (μαζι και του CJA) που καθορίζει αυτόματα το μέγεθος του συνόλου συμπύκνωσης (δηλαδή είναι μη παραμετρικός). Κατά συνέπεια, ο RSP3 εξαλείφει και τα δύο αδύνατα σημεία του CJA. Αξίζει να σημειωθεί ότι όπως οι CJA, RSP1 και RSP2, Το σύνολο συμπύκνωσης που κατασκευάστηκε από το RSP3 δεν εξαρτάται από τη σειρά δεδομένων στο σύνολο εκπαίδευσης.

Ψευδοκώδικας RSP3 [1]
Input: TS
Output: CS
1: $S \leftarrow \varnothing$
2: add(S, TS)
3: $CS \leftarrow \varnothing$
4: **repeat**
5: $C \leftarrow$ select the subset $\in S$ with the highest spli□ing criterion value
6: **if** C is homogeneous **then**
7: $r \leftarrow$ calculate the mean item by averaging the items in C
8: $r.label \leftarrow$ class of items in C
9: $CS \leftarrow CS \cup \{r\}$
10: **else**
11: $(D1, D2) \leftarrow$ divide C into two subsets
12: add($S, D1$)
13: add($S, D2$)
14: remove(S, C)
15: **end if**
16: **until** IsEmpty(S)
17: **return** CS

Ο παραπάνω ψευδοκώδικας παραθέτει τον RSP3. Χρησιμοποιεί μια απλή δομή δεδομένων S για να κρατήσει τα μη επεξεργασμένα υποσύνολα. Αρχικά, ολόκληρο το σύνολο εκπαίδευσης (TS) είναι ένα μη επεξεργασμένο υποσύνολο και τοποθετείται στο S . Σε κάθε επανάληψη, ο RSP3 επιλέγει το υποσύνολο C με τον υψηλότερο βαθμό του κριτηρίου διαίρεσης και ελέγχει εάν το C είναι ομοιογενές ή όχι. Εάν είναι ομοιογενές, το μέσο αντικείμενο υπολογίζεται με τον μέσο όρο των αντικείμενων στο C και τοποθετείται στο σύνολο συμπύκνωσης (CS) ως πρωτότυπο. Διαφορετικά, το C χωρίζεται σε δύο υποσύνολα D1 και D2 όπως θα έκανε και ο CJA. Αυτά τα νέα υποσύνολα προστίθενται στο S και το C αφαιρείται από το S. Η επανάληψη

συνεχίζεται έως ότου το S γίνει κενό, δηλαδή όλα τα υποσύνολα γίνουν ομογενή.

Λαμβάνοντας υπόψη τον RSP3, παρατηρούμε ότι παράγει λίγα πρωτότυπα για την εκπροσώπηση των περιοχών που δεν είναι κοντά στα σύνορα των κλάσεων και πολλά πρωτότυπα για την εκπροσώπηση των περιοχών που είναι κοντά στα σύνορα των κλάσεων. Βεβαίως, ο ρυθμός μείωσης που επιτυγχάνεται με το RSP3 εξαρτάται σε μεγάλο βαθμό από το επίπεδο θορύβου στα δεδομένα. Όσο υψηλότερο είναι το επίπεδο θορύβου στα δεδομένα, τόσο μικρότερα υποσύνολα δημιουργούνται και, κατά συνέπεια, τόσο χαμηλότερος ο ρυθμός μείωσης που επιτυγχάνεται. Αξίζει να σημειωθεί ότι η εύρεση των πιο απομακρυσμένων αντικειμένων σε κάθε υποσύνολο συνεπάγεται τους υπολογισμούς όλων των αποστάσεων μεταξύ των αντικειμένων του υποσυνόλου. Ως εκ τούτου, είναι δαπανηρές διαδικασίες που επιδεινώνουν το συνολικό κόστος προ-επεξεργασίας του αλγορίθμου. Σε περιπτώσεις μεγάλων συνόλων δεδομένων, Αυτό το μειονέκτημα μπορεί να καταστήσει την εκτέλεσή του απαγορευτική.

Κεφάλαιο 3ο: Κατηγοριοποίηση εγγυτέρων γειτόνων μέσω της βιβλιοθήκης scikit-learn της Python

3.1 Εισαγωγή

Σε αυτήν την ενότητα θα δούμε μερικά βασικά στοιχεία για την γλώσσα προγραμματισμού Python [55] που χρησιμοποιήσαμε στην παρούσα πτυχιακή εργασία και για την βιβλιοθήκη scikit-learn [56] που μέσα από αυτήν υλοποιήσουμε τα πειράματα. Θα εξηγήσουμε αναλυτικά τις κλάσεις και τις συναρτήσεις που χρησιμοποιήσαμε στα πειράματα, καθώς και έννοιες που θα τα κάνουν καλύτερα κατανοητά. Πιο συγκεκριμένα θα δούμε τις κλάσεις **neighbors**, η οποία είναι η βασική κλάση για κατηγοριοποίηση εγγυτέρων γειτόνων. Την κλάση **preprocessing**, η οποία είναι υπεύθυνη για διεργασίες προ-επεξεργασίας. Την κλάση **metrics**, η οποία μας δίνει διάφορες μετρικές για την απόδοση της κατηγοριοποίησης. Και της κλάσης **model_selection**, μέσω της οποίας διαλέγουμε κάποιο μοντέλο για την αποτίμηση της ακρίβειας. Για ευκολία θα έχουμε με έντονα γράμματά τις κλάσεις και πλάγια τις συνάρτησης.

3.2 Python

Η Python [55] είναι μια διερμηνευόμενη (interpreted) γλώσσα προγραμματισμού γενικού σκοπού (general-purpose) και υψηλού επιπέδου (high-level). Η φιλοσοφία σχεδιασμού της Python δίνει έμφαση στην αναγνωσιμότητα του κώδικα. Η δομή της γλώσσας καθώς και η αντικειμενοστραφής προσέγγιση της στοχεύουν να βοηθήσουν τους προγραμματιστές να γράψουν σαφή, λογικό κώδικα για έργα μικρής ή και μεγάλης κλίμακας. Είναι μια δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (garbage collector).

Σχεδιάστηκε από τον Ολλανδό προγραμματιστή Guido van Rossum ο οποίος άρχισε να εργάζεται στην Python στα τέλη της δεκαετίας του 80, ως διάδοχος της γλώσσας προγραμματισμού ABC, και κυκλοφόρησε για πρώτη φορά το 1991 ως Python 0.9.0 [57]. Η Python 2.0 κυκλοφόρησε το 2000 και εισήγαγε νέα χαρακτηριστικά, όπως κατανόηση λίστας και ένα σύστημα συλλογής απορριμμάτων χρησιμοποιώντας την καταμέτρηση αναφοράς. Η Python 3.0, την οποία χρησιμοποιούμε στην παρούσα πτυχιακή εργασία, κυκλοφόρησε το 2008 και ήταν μια σημαντική αναθεώρηση της γλώσσας που δεν είναι πλήρως συμβατή με τις προηγούμενες εκδόσεις της και αρκετός κώδικας Python 2 δεν εκτελείται χωρίς τροποποίηση στην Python 3. Η Python 2 διακόπηκε με την έκδοση 2.7.18 το 2020. Η Python πλέον χαρακτηρίζεται ως μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού.

Αντί να έχει όλη τη λειτουργικότητά της ενσωματωμένη στον πυρήνα της, η Python σχεδιάστηκε για να είναι εξαιρετικά επεκτάσιμη. Η Python επιδιώκει μια απλούστερη σύνταξη και γραμματική, ενώ παράλληλα δίνει στους προγραμματιστές επιλογές στη μεθοδολογία κωδικοποίησης. Η Python διαθέτει μια τεράστια πρότυπη βιβλιοθήκη (library), που συνήθως αναφέρεται ως μία

από τις μεγαλύτερες δυνάμεις της [58]. Εμείς θα αναφέρουμε μόνο όσες βιβλιοθήκες χρειάστηκαν για τα πειράματα της παρούσας πτυχιακής εργασίας.

3.3 Scikit-learn

Το Scikit-learn [56] γνωστό αλλιώς και ως sklearn είναι μια δωρεάν και ανοικτού λογισμικού βιβλιοθήκη της γλωσσάς Python η οποία επικεντρώνεται σε ζητήματα της εξόρυξης πληροφορίας και της μηχανικής μάθησης. Περιέχει διάφορους αλγορίθμους κατηγοριοποίησης, συσταδοποίησης και παλινδρόμησης οπως ο κατηγοριοποιητης k-εγγυτέρων γειτόνων, ο k-means, τα support-vector machines (SVM), ο DBSCAN και άλλοι πολλοί. Το scikit-learn σχεδιάστηκε για να λειτουργεί μαζί με τις αριθμητική βιβλιοθήκη NumPy και την επιστημονική βιβλιοθήκη SciPy της Python.

Το Scikit-learn ξεκίνησε με το όνομα scikits.learn ως ένα έργο του Γάλλου David Cournapeau για το Google Summer of Code to 2007. Το όνομα του βγαίνει από το "SciPy Toolkit" και υποδεικνύει ότι είναι μια προέκταση του SciPy. Τον Φεβρουάριο του 2010 προγραμματιστές από το Γαλλικό Εθνικό Ινστιτούτο Έρευνας στην Πληροφορική και τον Αυτοματισμό πήραν την ηγεσία του Scikit-learn, ξαναέγραψαν τον πηγαίο κωδικά και έκαναν την πρώτη δημόσια έκδοση του [59]. Το Scikit-learn είναι κυρίως γραμμένο σε Python, και χρησιμοποιεί την βιβλιοθήκη NumPy λόγο της υψηλής της απόδοσης στην γραμμική άλγεβρα και τις διεργασίες με πίνακες. Κάποιοι από τους βασικούς αλγορίθμους είναι γραμμένοι σε Cython, δηλαδή μια ανάμιξη της C++ και της Python.

Το Scikit-learn θεωρείται ως μια από τις πιο δημοφιλής βιβλιοθήκες μηχανικής μάθησης στον κόσμο. Και συνεργάζεται με αρμονία με άλλες βιβλιοθήκες της Python όπως, το Matplotlib και το plotly για σχεδιασμό, το Numpy για διανυσματοποίηση πινάκων, το Pandas χρησιμοποιώντας τα Dataframes του, το SciPy και άλλες πολλές ακόμα βιβλιοθήκες.

3.4 Sklearn.neighbors

Το **sklearn.neighbors** [60] είναι η κλάση του scikit-learn οπού μας επιτρέπει να κάνουμε πολύ εύκολα διεργασίες όπως αυτή των k-εγγυτέρων γειτόνων στην Python. Παρέχει λειτουργικότητα για μεθόδους μάθησης με βάση τους γείτονες με ή και χωρίς επίβλεψη. Η μέθοδος των εγγυτέρων γειτόνων χωρίς επίβλεψη είναι το θεμέλιο πολλών άλλων μεθόδων μάθησης, κυρίως πολλαπλής μάθησης (manifold learning) και φασματικής συσταδοποιησης (spectral clustering). Η εποπτευόμενη μάθηση που βασίζεται σε γείτονες χωρίζεται σε δυο μέρη, την κατηγοριοποίηση για δεδομένα με διακριτές ετικέτες και την παλινδρόμηση για δεδομένα με συνεχείς ετικέτες.

Η αρχή πίσω από τις μεθόδους εγγυτέρων γειτόνων είναι να βρούμε έναν προκαθορισμένο αριθμό αντικειμένων εκπαίδευσης που βρίσκονται πλησιέστερα σε απόσταση από το νέο αντικείμενο και να προβλέψουμε την ετικέτα με βάση αυτά. Το **sklearn.neighbors** μας επιτρέπει

να επιλέξουμε το πως θα επιλέγονται οι γείτονες. Ο αριθμός των γειτόνων μπορεί να είναι μια σταθερά που καθορίζεται από τον χρήστη (μάθηση k-εγγυτέρων γειτόνων) χρησιμοποιώντας την κλάση **KNeighborsClassifier** οπού το k είναι ένας ακέραιος αριθμός που καθορίζεται ως η σταθερά. Ή μπορεί να ποικίλλει ανάλογα με την τοπική πυκνότητα των αντικείμενα (μάθηση γειτόνων με βάση την ακτίνα) χρησιμοποιώντας την κλάση **RadiusNeighborsClassifier** οπού βρίσκει τους γείτονες σε μια προκαθορισμένη τιμή κινητής υποδιαστολής r, η οποία είναι η ακτίνα ενός αντικείμενου.

Ο κατηγοριοποιητης k-NN, τον οποίο και χρησιμοποιούμε εμείς, είναι ο πιο δημοφιλής από τους δυο. Όπως είπαμε η τιμή του k εξαρτάται σε τεράστιο βαθμό από τα δεδομένα. Σε περιπτώσεις οπού τα δεδομένα δεν είναι ομοιόμορφα κατανεμημένα ο κατηγοριοποιητης με βάση την ακτίνα μπορεί να είναι καλύτερη επιλογή, έτσι ώστε σε πιο αραιές γειτονιές να χρειάζονται λιγότεροι γείτονες. Όμως αυτή η μέθοδος επηρεάζεται σε πολύ μεγάλο βαθμό από την κατάρα των διαστάσεων.

Οι κλάσεις μεσα στο **sklearn.neighbors** μπορούν να διαχειριστούν είτε πίνακές από το Numpy είτε πίνακες απο το scipy.sparce ως είσοδο. Για πυκνούς πίνακες υπάρχουν πολλές διαθέσιμες μετρικές που υποστηρίζονται. Για αραιούς πίνακες υποστηρίζονται αυθαίρετες μετρικές Minkowski για τις αναζητήσεις.

Η κλάση **NearestNeighbors** του **sklearn.neighbors** εφαρμόζει την μη εποπτευόμενη μάθηση εγγυτέρων γειτόνων. Λειτουργεί ως μια ομοιόμορφη διεπαφή για τρεις διαφορετικούς αλγόριθμους εγγυτέρων γειτόνων: ball-tree, kd-tree και αλγόριθμος brute-force με βάση ρουτίνες στην κλάση **sklearn.metrics.pairwise**. Η επιλογή του αλγορίθμου ελέγχεται με την παράμετρο 'algorithm' και πρέπει να έχει μια από τις παρακάτω τιμές ('auto', 'ball_tree', 'kd_tree', 'brute'). Η προεπιλεγμένη τιμή είναι το 'auto' και ο αλγόριθμός προσπαθεί να εκτιμήσει την καλύτερη προσέγγιση εξετάζοντας κάποια χαρακτηριστικά από το σύνολο εκπαίδευσης.

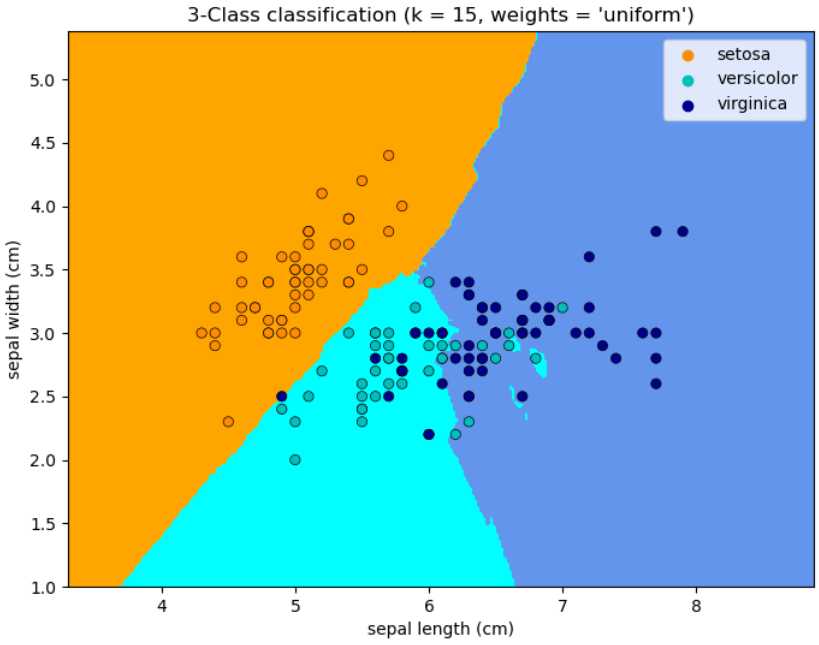
Στην περίπτωση που επιλέγει το 'brute' χρησιμοποιείται εξαντλητική αναζήτησή, δηλαδή ψάχνει τις αποστάσεις μεταξύ όλων των ζευγαριών που σχηματίζουν τα αντικείμενα ερωτημάτων με τα αντικείμενα του συνόλου εκπαίδευσης. Όταν έχουμε μικρά σύνολα δεδομένων η εξαντλητική αναζήτησή είναι πολύ ανταγωνιστική άλλα η απόδοση της πέφτει όσο μεγαλώνει το σύνολο.

Για να αντιμετωπιστούν οι αδυναμίες της εξαντλητικής αναζήτησης το scikit-learn χρησιμοποιεί μεθόδους δεικτοδότησης πολυδιάστατων δεδομένων με δενδρική δομή ώστε να μειωθούν οι αποστάσεις που υπολογίζονται. Με την λογική ότι αν το αντικείμενο A είναι μακριά από το αντικείμενο B και το B είναι κοντά στο αντικείμενο C τότε το A είναι μακριά από το C. Οι μέθοδοί δεικτοδότησης είναι αυτές που είδαμε στο προηγούμενο κεφάλαιο, δηλαδή kd-tree και ball-tree. Το scikit-learn εδώ μας επισημάνει ότι το kd-tree είναι αποδοτικό για όσο $D < 20$. Για να αντιμετωπιστούν οι αδυναμίες του kd-tree σε υψηλότερες διαστάσεις το scikit-learn υλοποιεί επίσης το ball-tree. Οι μέθοδοι δεικτοδότησης εναλλακτικά μπορούν να χρησιμοποιη-

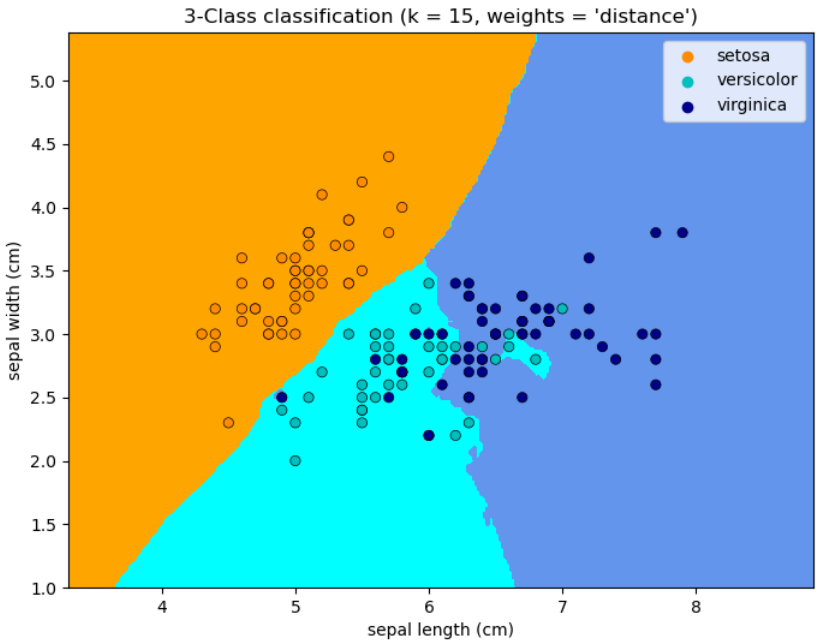
θούν και μέσω τον κλάσεων **KDTree** και **BallTree** οι οποίες κατασκευάστηκαν από τον Jake Vanderplas [61] το 2013, ο οποίος βελτίωσε σε απόδοση τις προηγούμενες εκδόσεις που υπήρχαν στο scikit-learn με την λογική ότι οι δενδρικές δομές θα γίνουν πιο γρήγορες αν τα αντικείμενα αποθηκεύονται μόνο στους κόμβους φύλλα. Και έτσι αποθηκεύουν ένα σύνολο αντικειμένων, στο οποίο θα γίνεται εσωτερική εξαντλητική αναζήτηση ώστε να επισκεπτόμαστε λιγότερους άλλους κόμβους.

Στο scikit-learn σε περίπτωση ισοπαλίας στην κατηγοριοποίηση κερδίζει η ετικέτα του πιο κοντινού γείτονα από τις συγκεκριμένες ετικέτες. Σε περίπτωση που οι δυο εγγύτεροι γείτονες έχουν ακριβώς την ίδια απόσταση αλλά διαφορετικές ετικέτες το αποτέλεσμα θα αποφασιστεί από την σειρά των δεδομένων στο σύνολο εκπαίδευσης [60].

Οι βασικοί κατηγοριοποιητές εγγυτέρων γειτόνων χρησιμοποιούν ομοιόμορφα βάρη, αυτό σημαίνει ότι όλοι οι εγγύτεροι γείτονες συνεισφέρουν το ίδιο στην κατηγοριοποίηση. Σε μερικές περιπτώσεις όμως μπορεί να είναι καλύτερα μπορεί να υπάρχει διάφορα στα βάρη των γειτόνων, για παράδειγμα οι πιο κοντινοί γείτονές να συνεφέρουν περισσότερο από τους μακρινούς εγγυτέρους γείτονες. Στο scikit-learn αυτό γίνετε μέσω της παραμέτρου weights. Η προεπιλεγμένη τιμή είναι το 'uniform', η οποία χρησιμοποιεί τα ομοιόμορφα βάρη. Όταν θέτουμε την παράμετρο ίση με 'distance' τότε το βάρος του κάθε γείτονα είναι αντιστρόφως ανάλογο με την απόσταση. Τα σχήματα 3.1 και 3.2 δείχνουν τους χάρτες κατηγοριοποιήσεις 15-NN για το πολύ γνωστό σύνολο Iris και είναι τα παραδείγματα του scikit-learn για την χρήση βαρών, όπως βλέπουμε υπάρχουν αρκετές διάφορες στα σύνορα των κλάσεων.



Σχήμα 3.1: Χάρτης κατηγοριοποίηση 15-NN χωρίς βάρη [5]



Σχήμα 3.2: Χάρτης κατηγοριοποίηση 15-NN με βάρη [5]

Για να αλλάξουμε την μετρική που θα τρέχει ο αλγόριθμος, χρησιμοποιούμε τις παραμέτρους 'metric' και 'p'. Το 'p' είναι παράμετρος για την μετρική Minkowski. Όταν είναι p=1 τότε είναι ίση με την απόσταση Manhattan (l1). Όταν είναι p=2 είναι ίση με την ευκλείδεια απόσταση (l2). Η παράμετρος 'metric' είναι η απόσταση που θα χρησιμοποιήσουμε. Η προεπιλεγμένες τιμές είναι η μετρική Minkowski και p=2 και αυτό συνεπάγεται με την ευκλείδεια απόσταση. Επιπλέον μπορούμε να χρησιμοποιήσουμε την παράμετρο 'metric_params', για επιπλέον πληροφορίες της μετρικής όμως η προεπιλεγμένη τιμή είναι 'None' και δεν είναι απαραίτητη.

Για να κλείσουμε με τις παραμέτρους του **KNeighborsClassifier** υπάρχουν ακόμα οι εξής :

- 'n_neighbors' που είναι ο αριθμός των γειτόνων που χρησιμοποιούμε στον κατηγοριοποιητή. Με προεπιλεγμένη τιμή το 5.

- 'n_jobs' που είναι ο αριθμός των παράλληλων διεργασιών που τρέχουν για την αναζήτηση των γειτόνων. Η προεπιλεγμένη τιμή είναι το 'None' που σημαίνει 1. Όταν έχουμε την τιμή '-1' χρησιμοποιούμε όλους τους πυρήνες.

- 'leaf_size' που είναι το μέγεθος των κόμβων φυλών στο kd-tree και ball-tree. Επειδή το brute force είναι πιο αποτελεσματικό σε μικρά σύνολα, οι δεντρικές δομές αλλάζουν εσωτερικά σε brute force όταν φτάνουν σε κόμβο φύλλο. Αυτό επηρεάζει την ταχύτητα αναζήτησης, κατασκευής του δέντρου και την μνήμη που χρειάζεται για την αποθήκευση του, όμως δεν επηρεάζει τα αποτελέσματα της αναζήτησης. Η βέλτιστη τιμή εξαρτάται από την φύση του προβλήματος, η προεπιλεγμένη τιμή είναι το 30 και ο αριθμός των αντικείμενων είναι $leaf\_size \leq n\_points \leq 2 * leaf\_size$.

Τώρα που εξηγήσαμε όλες τις παραμέτρους θα αναλύσουμε πως, επιλέγεται ο αλγόριθμος όταν η παράμετρος 'algorithm' έχει την τιμή 'auto'. Η επιλογή αυτή επηρεάζεται από τον αριθμό τον αντικειμένων N και την διάσταση D. Για μικρά σύνολα το brute force μπορεί να είναι πιο αποδοτικό από τις μεθόδους δεικτοδότησης. Ένας άλλος παράγοντας που επηρεάζει την επιλογή αυτή είναι η δομή των δεδομένων. Πιο συγκεκριμένα η εγγενής διάσταση και το ποσό αραιό είναι το σύνολο δεδομένων, δηλαδή το βαθμό στον οποίο τα δεδομένα γεμίζουν τον χώρο. Εδώ το scikit-learn μας επισημάνει ότι πρέπει να διαχωρίσουμε την έννοιά των αραιών δεδομένων από τις αραιές μήτρες. Γιατί τα δεδομένα μπορεί να μην έχουν μηδενικές τιμές άλλα να θεωρούνται πάλι αραιά. Επίσης ένας παράγοντας είναι η τιμή k, δηλαδή πόσους γείτονες αναζητούμε. Ο χρόνος αναζήτησης του brute force είναι σε μεγάλο βαθμό ανεπηρέαστος από αυτόν τον παράγοντα, ενώ τα οι μέθοδοι με δενδρική δομή γίνονται πιο αργοί όσο μεγαλώνει το k, επειδή ένα μεγάλο k οδηγεί στην ανάγκη αναζήτησης ενός μεγαλύτερου μέρους του χώρου και δεύτερον, η χρήση ενός k > 1 απαιτεί εσωτερική ουρά αποτελεσμάτων καθώς το δέντρο διασχίζεται.

Το 'auto' επιλέγει την τιμή 'brute' εάν είναι αληθής κάποια από τις παρακάτω υποθέσεις.

- Τα δεδομένα εισόδου είναι αραιά.
- η παράμετρος 'metric' ειναι ιση με 'precomputed'
- $D > 15$
- $k \geq N/2$
- Το 'effective_metric_' (Το metric που έχουμε διαλέξει ως παράμετρο) δεν υπάρχει στο 'VALID_METRICS' (διαθέσιμες μετρικές) για το kd-tree ή το ball-tree.

Αλλιώς επιλέγει το πρώτο από τα kd-tree και ball-tree που έχουν το 'effective_metric_' ως 'VALID_METRICS'.

3.5 Προ-επεξεργασία μέσω scikit-learn

Σχεδόν όλες οι διεργασίες εξόρυξης δεδομένων και μηχανικής μάθησης χρειάζονται ένα είδος προ-επεξεργασίας. Στο scikit-learn αυτό γίνεται μέσω της κλάσης **sklearn.preprocessing** [62] το οποίο παρέχει πολλές χρήσιμες λειτουργίες και κλάσεις μετασχηματιστών ώστε να αλλάξει τα ακατέργαστα δεδομένα σε μια αναπαράσταση καταλληλότερη για τις διεργασίες μας.

Εμείς χρησιμοποιήσαμε μόνο την κλάση **MinMaxScaler**. Η λειτουργία της είναι αυτό που περιγράψαμε ως κανονικοποίηση στο κεφάλαιο ένα με τον τύπο 1.2. Ένας άλλος χρήσιμος μετασχηματιστής είναι ο **StandardScaler** ο οποίος αφαιρεί το μέσο ορό και κλιμακώνει με βάση την διακύμανση. Η τυποποίηση ενός συνόλου δεδομένων είναι μια κοινή απαίτηση για

πολλούς εκτιμητές μηχανικής μάθησης οι οποίοι ενδέχεται να παρουσιάσουν προβλήματα εάν τα χαρακτηριστικά δεν μοιάζουν με τυπικά κανονικά κατανεμημένα δεδομένα. Ακόμα ένας χρήσιμος μετασχηματιστής είναι ο **MaxAbsScaler** ο οποίος κλιμακώνει κάθε χαρακτηριστικό έτσι ώστε η μέγιστη απόλυτη τιμή αυτών να είναι 1, δηλαδή η τιμή του θα βρίσκεται μεταξύ -1 και 1.

Στην περίπτωση που έχουμε ονομαστικά χαρακτηριστικά για παράδειγμα φύλλο με τιμές [Άντρας,Γυναίκα]. Μπορούμε να μετατρέψουμε τέτοιες τιμές σε ακέραιες δηλαδή [0,1]. Αυτό γίνεται δυνατό μέσα από την κλάση **OrdinalEncoder**. Ένας άλλος μετασχηματιστής που προσφέρεται είναι μέσα από την κλάση **OneHotEncoder** οπού μια τιμή μετασχηματίζεται σε 1 και όλες οι άλλες σε 0, για παράδειγμα το χαρακτηριστικό χώρα με τιμές [Ελλάδα,Ιταλία,Ισπανία] μετατρέπεται σε [1,0,0].

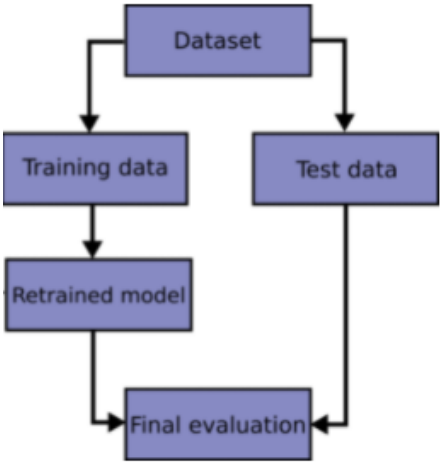
3.6 Απόδοση κατηγοριοποίησης

Για να εξετάσουμε πόσο έμπιστος είναι ένας κατηγοριοποιητης χρειαζόμαστε κάποια κριτήρια. Το scikit-learn μας δίνει αυτήν την δυνατότητα μέσα από την κλάση **sklearn.metrics** [63] (δεν πρέπει να το συσχετίσουμε τις μετρικές τις απόστασης που συζητήσαμε μέχρι τώρα). Η μετρική που χρησιμοποιούμε εμείς είναι η ακρίβεια, η οποία δύνεται από τον τύπο $A = C/T$ οπού το C είναι ο αριθμός των σωστών προβλέψεων και το T είναι ο συνολικός αριθμός προβλέψεων του κατηγοριοποίηση. Μπορούμε πολύ εύκολα να βρούμε αυτήν την τιμή από την συνάρτηση *accuracy_score*, η οποία έχει ως εισόδους τις πραγματικές ετικέτες του συνόλου έλεγχου και τις ετικέτες που πρόβλεψε ο κατηγοριοποιητης και να τις συγκρίνει.

Εάν θέλουμε να ξέρουμε περισσότερες πληροφορίες για τα αποτελέσματα της κατηγοριοποίησης μας δίνετε αυτή η δυνατότητα μέσα από την συνάρτηση *classification_report*. Η οποία πέρα από την ακρίβεια μας παρουσιάζει για κάθε κλάση του συνόλου ξεχωριστά την ορθότητα (precision), δηλαδή ο λόγος μεταξύ του πλήθους των σωστών προβλέψεων της κλάσης και του πλήθους συνολικών προβλέψεων της κλάσης, και την ευαισθησία (recall), δηλαδή ο λόγος μεταξύ του πλήθους των σωστών προβλέψεων της κλάσης και του πραγματικού πλήθους της κλάσης. Επίσης μας δείχνει και την τιμή support, δηλαδή το πραγματικό πλήθος της κάθε κλάσης, και της τιμής f1-score το οποίο βγαίνει από τον τύπο $F1 = 2 * \frac{(precision * recall)}{(precision + recall)}$.

3.7 Μέθοδοι αποτίμησης της ακρίβειας

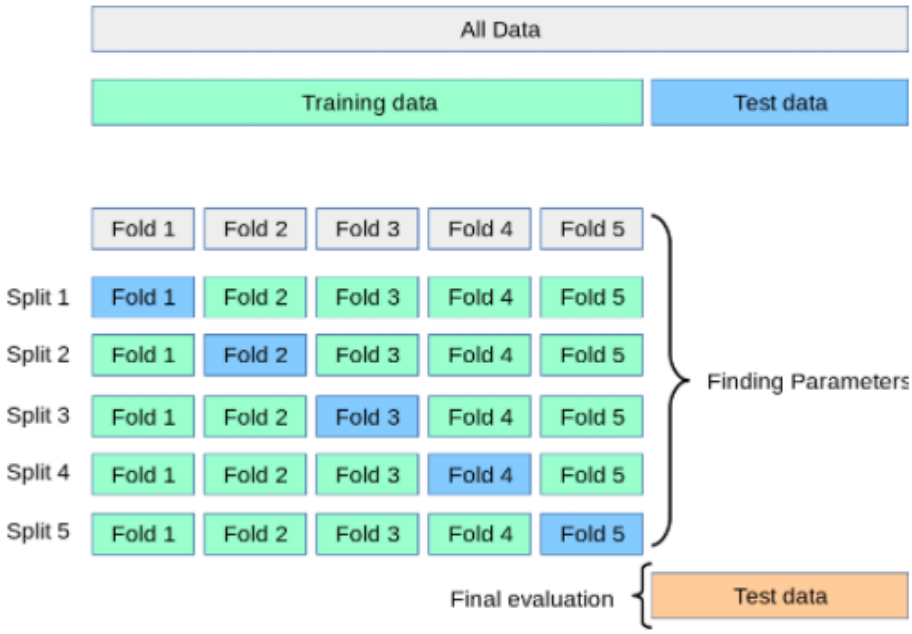
Αρκετοί τρόποι είναι διαθέσιμοι για να μπορέσουμε να αποτιμήσουμε την ακρίβεια κατηγοριοποίησης, το scikit-learn μας προσφέρει αρκετούς από αυτούς, τους οποίους και θα αναλύσουμε παρακάτω, μέσω της κλάσης **sklearn.model_selection** [6]. Ο πιο εύκολος από αυτούς ονομάζεται Hold out, κατά τον οποίο χωρίζουμε το σύνολο δεδομένων σε δυο τμήματα, πρώτον το σύνολο εκπαίδευσης (training set) π.χ. 2/3 των δεδομένων και δεύτερον το σύνολο ελέγχου



Σχήμα 3.3: Κατηγοριοποίηση μέσω train test split [6]

(testing set) π.χ. 1/3 των δεδομένων. Έτσι δημιουργούμε ένα μοντέλο κατηγοριοποίησης με το σύνολο εκπαίδευσης και κατηγοριοποιούμε κάθε αντικείμενο του σύνολο έλεγχου. Η μέτρηση της ακρίβειας καθορίζεται από τον τύπο $A = X/N$, όπου το X είναι ο αριθμός των αντικειμένων που κατηγοριοποιούνται σωστά και N είναι ο αριθμός αντικείμενων στο σύνολο έλεγχου. Το μειονέκτημα είναι ότι η ακρίβεια εξαρτάται από την διάταξη των αντικειμένων.

Μια παραλλαγή του Hold out ειναι το random subsampling. Εδώ για να αποφύγουμε την εξάρτηση από την διάταξη εφαρμόζουμε τυχαία δειγματοληψία, επιλέγοντάς N τυχαία αντικείμενα τα οποία έχουμε ως σύνολο έλεγχου και όλα τα υπόλοιπα γίνονται το σύνολο εκπαίδευσης. το μειονέκτημα εδώ είναι ότι έχουμε διαφορετικά αποτελέσματα κάθε φορά που τρέχουμε τον αλγόριθμο και για αυτό το λόγο η ακρίβεια είναι ο μέσος όρος και βγαίνει από τον τύπο



Σχήμα 3.4: Κατηγοριοποίηση μέσω 5 Fold cross validation [6]

$A = \sum_{i=1}^{k} \frac{X_i}{N}$ Όπου X_i είναι ο αριθμός των σωστά κατηγοριοποιημένων αντικείμενων στην i-στη επανάληψη. Και τους δυο παραπάνω τρόπους μπορούμε να τους χρησιμοποιήσουμε στο scikit-learn μέσα από την συνάρτηση *train_test_split*. Όταν στην παράμετρο της συνάρτησης 'random_state' έχουμε τιμή ίση με 'None' που είναι και η προεπιλεγμένη τότε χρησιμοποιούμε το random subsampling. Όταν έχουμε οποιοδήποτε ακέραιο ως τιμή τότε χρησιμοποιούμε το Hold out. Στο σχήμα 3.3 παρουσιάζεται αυτή η διαδικασία.

Ένας ακόμα πολύ γνωστός τρόπος αποτίμησης είναι το K-Fold cross validation τον οποίο χρησιμοποιούμε για να αποφύγουμε την τυχαία δειγματοληψία. Έστω M ο αριθμός των αντικειμένων σε ένα σύνολο δεδομένων και K ο αριθμός των επαναλήψεων. Χωρίζουμε το σύνολο σε K τμήματα με M/K αντικείμενα το κάθε ενα. Στην i-στη επανάληψη, το i-στο τμήμα λειτουργεί ως το σύνολο έλεγχου, ενώ όλα τα υπόλοιπα $K - 1$ τμήματα παίρνουν το ρόλο του συνόλου εκπαίδευσης. Μετρώντας στο τέλος το μέσο όρο ακρίβειας από τις επαναλήψεις. Στο scikit-learn μπορούμε να του χρησιμοποιήσουμε αυτή την μέθοδο μέσα από την υποκλάση **KFold** ή με μεγαλύτερη ευκολία μέσα από την συνάρτηση του **sklearn.model_selection**, *cross_validate* επιλέγοντας τον αριθμό K ως παράμετρο 'cv'. Επίσης υπάρχουν οι συναρτήσεις *cross_val_test* και *cross_val_predict* που κάνουν την ίδια δουλεία άλλα έχουν διαφορετικές εξόδους. Αυτή η διαδικασία παρουσιάζεται στο σχήμα 3.4.

Κεφάλαιο 4ο: Πειραματική Μελέτη

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα εξετάσουμε και θα συγκρίνουμε τις μεθόδους δεικτοδότησης, δηλαδή kd-tree και ball-tree, με τις ΤΜΔ, δηλαδή CNN, IB2, RSP3, ENN. Θα χωρίσουμε το πείραμα σε τρία μέρη ανάλογα με το είδος της προ-επεξεργασίας και θα εκτελέσουμε τον αλγόριθμο K-NN με παραμέτρους k=5 και k=1 αφού πρώτα δούμε τα σύνολα δεδομένων του πειράματος.

Στο πρώτο μέρος θα εξεταστούν τα σύνολα δεδομένων χωρίς καμία προ-επεξεργασία πριν την εκτέλεση. Στο δεύτερο μέρος του πειράματος θα εξεταστούν οι ΤΜΔ αλλά χρησιμοποιώντας μόνο συμπύκνωση. Εδώ δεν χρησιμοποιούμε καμία από τις μεθόδους δεικτοδότησης αλλά μόνο το brute force. Στο τρίτο μέρος του πειράματος θα εξεταστούν οι ΤΜΔ άλλα χρησιμοποιώντας πρώτα αλγορίθμους επεξεργασίας για να φτιάξουμε σύνολα χωρίς θόρυβο. Αφού καταγράφουν όλες τις μετρήσεις σε πίνακες θα συγκρίνουμε τα αποτελέσματα. Έπειτα θα εκτελεστεί ένα δεύτερο πείραμα μέσω του προγράμματος Weka. Θα χρησιμοποιήσουμε επιλογή χαρακτηριστικών για να δούμε πως οι διάφορες διαστάσεις επηρεάζουν τις μεθόδους δεικτοδότησης. Και έτσι θα καταλήξουμε στο συμπέρασμα, ποια είναι η καλύτερη μέθοδος σε ένα σύνολο με δεδομένο αριθμό χαρακτηριστικών.

4.2 Συνολα δεδομενων

Τα σύνολα δεδομένων που χρειάζονταν για τα πειράματα πρέπει να πληρούν ορισμένα κριτήρια. Το πρώτο και πιο σημαντικό από αυτά, είναι σαφώς το ότι θα πρέπει να είναι εξειδικευμένα σύνολα δεδομένων τα οποία προορίζονται για θέματα κατηγοριοποίησης. Δηλαδή ένα πεδίο του συνόλου θα πρέπει να περιγράφει την κλάση στην οποία ανήκει το αντικείμενο. Θα πρέπει επίσης τα σύνολα να είναι διαφορετικού αριθμού χαρακτηριστικών, ώστε να εξετάσουμε τις

Πίνακας 4.1: Συνολα δεδομενων

Α/Α	Όνομά	Διαστάσεις	Μέγεθος	κλάσεις
1	Banana	2	5300	2
2	Skin	3	245057	2
3	Phoneme	5	5404	2
4	Shuttle	9	58000	7
5	MGT	10	19020	2
6	Poker	10	1025010	10
7	Wine	11	4898	11
8	Marketing	13	6876	9
9	PID	16	10992	10
10	LIR	16	20000	26
11	Ring	20	7200	2
12	KDD	36	141486	23

μεθόδους δεικτοδότησης και τις ΤΜΔ σε πολλές διαφορετικές συνθήκες.

Τα σύνολα δεδομένων μπορούν να είναι είτε δυαδικά (binary), να έχουν δηλαδή μόνο δύο κλάσεις, ή εναλλακτικά να έχουν περισσότερες κλάσεις (multiclass) και θέσαμε ένα κατώτατο όριο αντικειμένων σε κάθε σύνολο δεδομένων. Θα πρέπει όλα τα σύνολα να έχουν πάνω από 5000 αντικείμενά ώστε να μπορέσουμε να μετρήσουμε τον χρόνο, π.χ αν ένα σύνολο έχει μόνο 500 αντικείμενά και λόγο της ταχύτητάς της python ο χρόνος επεξεργασίας θα ήταν μερικά εκατοστά του δευτερολέπτου και δεν θα ήταν εύκολη η σύγκρισή και επίσης χρειαζόμαστε μεγαλύτερά σύνολα ώστε οι ΤΜΔ να μπορούν να αφαιρέσουν ένα πολύ μεγάλο ποσοστό και πάλι να έχουν απομείνει πολλά αντικείμενά.

Για να εκτελέσουμε τις ΤΜΔ καθώς και τον κατηγοριοποιητή εγγύτερων γειτόνων χρησιμοποουμε την Ευκλείδεια απόσταση, έτσι θα πρέπει όλα τα χαρακτηριστικά να είναι αριθμητικά. Αυτό, βοηθάει στο να αξιολογήσουμε τα αποτελέσματα μεταξύ των διαφορετικών συνόλων δεδομένων. ΄Οσα σύνολα δεδομένων έχουν ονομαστικά χαρακτηριστικά δυστυχώς δεν μπορούν να χρησιμοποιηθούν από τις ΤΜΔ, επειδή οι αυτές τεχνικές χρειάζονται κάποιο κριτήριο για να υπολογίζουν την απόστασή μεταξύ των αντικειμένων και στην περίπτωση μας αυτό το κριτήριο είναι η Ευκλείδεια απόσταση. Φυσικά, θα μπορούσαμε να πραγματοποιήσουμε κάποια μετατροπή πάνω σε ονομαστικά δεδομένα για να τους δώσουμε αριθμητική μορφή όπως το Label encoding του scikit-learn, όμως αυτή θα ήταν μία αναξιόπιστη λύση. Καθώς όπως θα δούμε από το σύνολο marketing, στο οποίο έγινε ακριβώς αυτό (από τους δημιουργούς του συνόλου), υποφέρει σε πάρα πολύ μεγάλο βαθμό η ακρίβειά της κατηγοριοποίησης.

Είναι απαραίτητο να έχουν όλα τα χαρακτηριστικά το ίδιο εύρος τιμών, καθώς θεωρούμε πως είναι όλα τους το ίδιο σημαντικά. Για να αντιμετωπιστεί αυτό το θέμα, πραγματοποιήθηκε μία διαδικασία κλιμάκωσης (scaling) με το min max scaler του scikit-learn έτσι ώστε να προσαρμοστούν οι τιμές ανάλογα. Μέσω της κλιμάκωσης, οι τιμές μεταβλήθηκαν έτσι ώστε όλα τα χαρακτηριστικά να αποτελούν πραγματικούς αριθμούς που κυμαίνονται στα ορια του μηδέν και του ένα (0, 1). ΄Ετσι δίνουμε σε όλα τα χαρακτηριστικά το ίδιο βάρος χωρίς να αλλοιωθεί η σημασία των τιμών τους.

Ακολουθεί μία σύντομη περιγραφή των συνόλων δεδομένων που χρησιμοποιήθηκαν στα πειράματα. Συνολικά χρησιμοποιήθηκαν δώδεκα σύνολα δεδομένων τα οποία έχουν διαφορετικό αριθμό διαστάσεων και αντικειμένων. Όλα τα σύνολα δεδομένων είναι διαθέσιμα στο KEEL [43] και το UCI Machine Learning Repository [64].

- Το σύνολο δεδομένων Poker είναι το μεγαλύτερο σύνολο που χρησιμοποιήθηκε στην παρούσα πτυχιακή εργασία . Κάθε εγγραφή αυτού του συνόλου δεδομένων είναι ένα παράδειγμα ενός χεριού Poker που αποτελείται από πέντε τραπουλόχαρτα που προέρχονται από μια τυπική τράπουλα των 52 φύλλων. Κάθε κάρτα περιγράφεται χρησιμοποιώντας δύο χαρακτηριστικά (είδος και αξία), φτιάχνοντάς ένα σύνολο 10 ονομαστικών χαρακτηριστικών.

Το χαρακτηριστικό που δείχνει σε ποια κλάση ανήκει το κάθε αντικείμενο, περιγράφει το χέρι πόκερ που λαμβάνεται:

0. τίποτα στο χέρι. Δεν αναγνωρίζεται χέρι πόκερ.
1. ένα ζευγάρι. Ένα ζευγάρι ίσης άξιας μέσα σε πέντε φύλλα.
2. δύο ζεύγη; Δύο ζεύγη ίσων άξιων μέσα σε πέντε φύλλα.
3. τρία από ένα είδος. Τρία φύλλα με ίση άξια μέσα σε πέντε φύλλα.
4. Κέντα. Πέντε φύλλα που κατατάσσονται διαδοχικά χωρίς κενά.
5. Χρώμα. Πέντε φύλλα με του ίδιου είδους.
6. Φουλ. Ένα ζεύγος και διαφορετικής άξιας τρία από ένα είδος.
7. Καρέ. Τέσσερά φύλλα ίσης άξιας μέσα σε πέντε φύλλα.
8. Κέντα Χρώμα. Πέντε φύλλα που κατατάσσονται διαδοχικά χωρίς κενά του ίδιου είδους.
9. Φλος Ρουαγιάλ. Άσσος, Ρήγας, Ντάμα, Τζακ, δέκα του ίδιου είδους.

Το πρώτο χαρακτηριστικό μας δείχνει το είδος του πρώτου φύλλου, το δεύτερο μας δείχνει την άξια του πρώτου φύλλου, Το τρίτο μας δείχνει το είδος του δεύτερου φύλλου, το τέταρτο μας δείχνει την άξια του δεύτερου φύλλου και συνεχίζουμε έτσι για τα πέντε φύλλα στο χέρι. Επίσης έχει 1025010 αντικείμενα.

- Το σύνολο KDD πρόκειται για ένα από τα πιο διαδεδομένα σύνολα στο κόσμο της εξόρυξης δεδομένων. Έχει πολλές εκδόσεις, αλλά η συγκεκριμένη που χρησιμοποιείται σε αυτήν την πτυχιακή εργασίας είναι το 10% του KDD Cup 1999 , η οποία δημιουργήθηκε για τον Τρίτο Διεθνή Διαγωνισμό Ανίχνευσης Γνώσης και Εργαλείων Εξόρυξης Δεδομένων (Third International Knowledge Discovery and Data Mining Tools Competition) η οποία πραγματοποιήθηκε σε συνδυασμό μαζί με το πέμπτο Διεθνές Συνέδριο Ανίχνευσης Γνώσης και Εργαλείων Εξόρυξης Δεδομένων. Σκοπός του διαγωνισμού ήταν η δημιουργία ενός ανιχνευτή εισβολής δικτύου, ενός προγνωστικού μοντέλου ικανού να ξεχωρίσει σε ένα δίκτυο τις καλές συνδέσεις από τις κακές, που ονομάζονται εισβολές ή επιθέσεις, χρησιμοποιώντας τόσο ονομαστικά όσο και συνεχή δεδομένα.

  Το δείγμα του KDD που χρησιμοποιήθηκε σε αυτήν την εργασία είναι ένα υποσύνολο του αρχικού, από το οποίο έχουν αφαιρεθεί τα διπλότυπα. Το πλήρες σύνολο δεδομένων περιέχει έναν τεράστιο αριθμό αντικειμένων. Το συγκεκριμένο υποσύνολο έχει 41 χαρακτηριστικά, από τα οποία χρησιμοποιούνται τα 36 αριθμητικά χαρακτηριστικά, το σύνολο αποτελείται από 23 κλάσεις και 141486 αντικείμενα.

- Το σύνολο Shuttle πρόκειται για ένα σύνολο δεδομένων με σκοπό να βοηθάει διαστημικά αεροσκάφη κατά την διάρκεια της προσγείωσης τους. Δημιουργήθηκε αρχικά για να

εξάγει κατανοητούς κανόνες για τον προσδιορισμό των συνθηκών υπό τις οποίες μια αυτόματη προσγείωση θα ήταν προτιμότερη από τον χειροκίνητο έλεγχο ενός διαστημικού οχήματος. Ο στόχος είναι να αποφασιστεί ποιος τύπος ελέγχου του σκάφους θα πρέπει να χρησιμοποιηθεί.

Το σύνολο δεδομένων Shuttle περιέχει 9 χαρακτηριστικά τα οποία είναι αριθμητικά. Υπάρχουν 7 πιθανές τιμές για την ετικέτα κλάσης: (Rad Flow, Fpv Close, Fpv Open, High, Bypass, Bpv Close, Bpv Open) και 58000 αντικείμενα.

- Το σύνολο Letter Image Recognition έχει στόχο να προσδιοριστεί ο καθένας από έναν μεγάλο αριθμό ασπρόμαυρων ορθογώνιων εικονοστοιχείων ως ένα από τα 26 κεφαλαία γράμματα στο αγγλικό αλφάβητο. Οι εικόνες χαρακτήρων βασίστηκαν σε 20 διαφορετικές γραμματοσειρές και κάθε γράμμα μέσα σε αυτές τις 20 γραμματοσειρές παραμορφώθηκε τυχαία για να παράγει ένα αρχείο 20.000 μοναδικών ερεθισμάτων. Κάθε ερέθισμα μετατράπηκε σε 16 αδιάσπαστα αριθμητικά χαρακτηριστικά (στατιστικές ροπές και μετρήσεις άκρων) τα οποία στη συνέχεια κλιμακώθηκαν ώστε να ταιριάζουν σε ένα εύρος ακεραίων τιμών από 0 έως 15.

- Το σύνολο MAGIC Gamma Telescope, αυτό το σύνολο δεδομένων περιέχει παραγόμενα δεδομένα για την προσομοίωση της καταγραφής σωματιδίων γάμμα υψηλής ενέργειας σε ένα επίγειο ατμοσφαιρικό τηλεσκόπιο Cherenkov gamma χρησιμοποιώντας την τεχνική απεικόνισης. Το σύνολο δεδομένων δημιουργήθηκε από ένα πρόγραμμα του Monte Carlo το 1998, που ονομάζετε Corsica.

  Ο στόχος είναι να διακρίνουμε στατιστικά τις εικόνες που παράγονται από τα πρωτεύοντα gammas (σήμα, ετικέτα κλάσης g) από τις εικόνες των hadronic showers που ξεκινούν από τις κοσμικές ακτίνες στην ανώτερη ατμόσφαιρα (φόντο, ετικέτα κλάσης h). Το σύνολο δεδομένων MAGIC Gamma Telescope περιέχει 10 χαρακτηριστικά τα οποία είναι τα εξης : (FWidth, FSize, FConc, FConc1, FAsym, FM3Long, FM3Trans, FAlpha, FDist). Υπάρχουν 2 πιθανές τιμές για την ετικέτα κλάσης: (g,h) και 19020 αντικείμενα.

- Το συνολο Pen-Based Recognition of Handwritten Digits, το οποίο είναι μια βάση δεδομένων ψηφίων που δημιουργήθηκε με τη συλλογή 250 δείγματα από 44 συγγραφείς, χρησιμοποιώντας μόνο (X, Y) πληροφορίες συντεταγμένων που αντιπροσωπεύονται ως φορείς χαρακτηριστικών σταθερού μήκους, τα οποία μετατράπηκαν σε 8 σημεία ανά ψηφίο ως εκ τούτου, το σύνολο δεδομένων περιέχει 8 σημεία επί 2 συντεταγμένες, δηλαδή 16 χαρακτηριστικά και περιεχέι 10992 αντικείμενα.

- Το σύνολο δεδομένων Marketing, Αυτό το σύνολο δεδομένων περιέχει ερωτήσεις από ερωτηματολόγια που συμπληρώθηκαν από πελάτες εμπορικού κέντρου στην περιοχή του

Κόλπου του Σαν Φρανσίσκο. Ο στόχος είναι να προβλεφθεί το ετήσιο εισόδημα των νοικοκυριών από τα άλλα 13 δημογραφικά χαρακτηριστικά.

Το χαρακτηριστικά του σύνολου έχουν μετατραπεί σε αριθμητικά χαρακτηριστικά ανά κατηγορίες των απαντήσεων και είναι τα εξής : (φύλο, οικογενειακή κατάσταση, ηλικία, επίπεδο εκπαίδευσης , επάγγελμα , χρονιά διαμονής στην περιοχή του σαν φραντσικο/ οακλαντ/ σαν χοσε, αν οι παντρεμένοι μοιράζονται το εισόδημα τους, άτομα που συγκατοικεί μαζί, άτομα που συγκατοικεί μαζί κάτω των 18 ετών, αν τους ανήκει το σπίτι που κατοικούν, τύπος του σπιτιού, εθνικότητα, ποια γλώσσα ομιλείται περισσότερο μέσα στο σπίτι). Υπάρχουν 9 κλάσεις και 6876 αντικείμενα.

- Το σύνολο Ringnorm, αυτό είναι ένα πρόβλημα κατηγοριοποίησης 2 κλάσεων και 20 διαστάσεων, Κάθε κλάση προέρχεται από μια πολυπαραγοντική κανονική κατανομή. Η κλάση 1 έχει μέση τιμή μηδέν και διακύμανσης 4 φορές την ταυτότητα. Η κλάση 2 έχει μέση τιμή (a, a,..a) και διακύμανση μονάδας. $a = 2/\sqrt{20}$. Το σύνολο περιέχει 7200 αντικείμενα.

- Το σύνολο Phoneme, ο στόχος αυτού του συνόλου δεδομένων είναι να γίνει διάκριση μεταξύ ρινικών (κλάσης 0) και προφορικών ήχων (κλάσης 1). Η κατανομή της κατηγορίας είναι 3.818 δείγματα στην κατηγορία 0 και 1.586 δείγματα στην κατηγορία 1.

  Περιεχέι 5 χαρακτηριστικά τα οποία είναι τα φωνήματα που μεταγράφονται ως εξής: sh όπως στο she, dcl όπως στο dark, iy ως το φωνήεν στο she, aa ως το φωνήεν στο dark και ao ως το πρώτο φωνήεν στο water. το σύνολο περιεχέι 5404 αντικείμενα.

- Το σύνολο Banana Ένα τεχνητό σύνολο δεδομένων όπου τα αντικείμενα του ανήκουν σε πολλές συστάδες με σχήμα μπανάνας.

  Υπάρχουν δύο χαρακτηριστικά At1 και At2 που αντιστοιχούν στον άξονα X και Y, αντίστοιχα. Η ετικέτα κλάσης (-1 και 1) αντιπροσωπεύει ένα από τα δύο σχήματα μπανάνας στο σύνολο δεδομένων. Το σύνολο περιεχέι 5300 αντικείμενα.

- Το σύνολο White Wine Quality, αυτο το σύνολο δεδομένων σχετίζεται με τη λευκή παραλλαγή του πορτογαλικού κρασιού Vinho Verde. Λόγω της ιδιωτικών και υλικοτεχνικών ζητημάτων, είναι διαθέσιμες μόνο φυσικοχημικές (εισροές) και αισθητηριακές (έξοδος) μεταβλητές (π.χ. δεν υπάρχουν δεδομένα σχετικά με τους τύπους σταφυλιών, το εμπορικό σήμα κρασιού, την τιμή πώλησης κρασιού κ.λπ.).

  Αυτά τα σύνολα δεδομένων μπορούν να θεωρηθούν ως διεργασίες κατηγοριοποίησης ή παλινδρόμησης. Οι κλάσεις είναι διατεταγμένες και μη ισορροπημένες (π.χ. υπάρχουν πολύ πιο φυσιολογικά κρασιά από τα εξαιρετικά ή τα φτωχά). Το συνολο περιέχει

τα εξής 11 χαρακτηριστικά: (FixedAcidity, VolatileAcidity, CitricAcid, ResidualSugar, Chlorides, FreeSulfurDioxide, TotalSulfurDioxide, Density, PH, Sulphates, Alcohol) και 11 κλάσεις που δείχνουν την ποιότητα του κρασιού. Το σύνολο περιεχέι 4898 αντικείμενα.

- Το σύνολο Skin , αυτο σύνολο δεδομένων συλλέχτηκε με τυχαία δειγματοληψία των τιμών B,G,R από εικόνες προσώπου διαφόρων ηλικιακών ομάδων (νέοι, μεσαίοι και ηλικιωμένοι), φυλετικές ομάδες (λευκοί, μαύροι και ασιάτες) και φύλα που λαμβάνονται από τη βάση δεδομένων FERET και τη βάση δεδομένων PAL. Το συνολικό μέγεθος δείγματος μάθησης είναι 245057. Από τα οποία 50859 είναι τα δείγματα δέρματος και 194198 είναι δείγματα μη δέρματος.

  Αυτό το σύνολο δεδομένων έχει 3 διαστάσεις όπου οι τρεις πρώτες στήλες είναι τιμές B,G,R (x1, x2 και x3 χαρακτηριστικά) και η τέταρτη στήλη είναι των ετικετών κλάσης (μεταβλητή y).

4.3 Πειραματικές Μετρήσεις

Σε αυτό το υποκεφάλαιο θα συγκρίνουμε τις μεθόδους δεικτοδότησης, δηλαδή kd-tree και ball-tree και τις ΤΜΔ, δηλαδή CNN, IB2, RSP3, ENN. Το πείραμα θα χωριστεί σε τρία μέρη ανάλογα με το είδος προ-επεξεργασίας που χρησιμοποιήθηκε δηλαδή, χωρίς προ-επεξεργασία, μόνο συμπύκνωση και επεξεργασία και συμπύκνωση. Ως μετρική χρησιμοποιήθηκε η ευκλείδεια απόσταση. Οι παράμετροι του **KNeighborsClassifier** εμειναν οι προεπιλεγμένοι γιατί παρατηρήθηκε ότι είναι και οι πιο αποδοτικοί. Τα σύνολα δεδομένων είναι ήδη έτοιμα σε μορφή 5-fold ή μετατράπηκαν σε αυτήν την μορφή πριν από το πείραμα. Χρησιμοποιήσαμε 5-fold cross validation, επειδή είναι μια αποτελεσματική μέθοδος αποτίμησης της ακρίβειας. Οι ΤΜΔ εφαρμόστηκαν μόνο στο σύνολο εκπαίδευσης του κάθε fold γιατί θέλουμε να εξετάσουμε την ακρίβεια κατηγοριοποίησης και τον χρόνο αναζήτησής χωρίς να αλλάξουμε τίποτα στα σύνολα έλεγχου. Οι παράμετροι k που χρησιμοποιήθηκαν είναι οι τιμές 5 και 1. Το 1 χρησιμοποιήθηκε γιατί μερικές φορές οι ΤΜΔ αδικούνται από μεγαλύτερες τιμές k ιδιαίτερα όταν πετυχαίνουν μεγάλη μείωση δεδομένων.

Στο πρώτο μέρος θα εξετάσουμε τα σύνολα δεδομένων χωρίς να κάνουμε καμία προ-επεξεργασία πριν την εκτέλεση. Πρώτα θα εκτελέσουμε τον αλγόριθμο k-NN με brute force και έπειτα με kd-tree και ball-tree έτσι θα δούμε πως συγκρίνονται στις διάφορες διαστάσεις.

Στο δεύτερο μέρος του πειράματος θα εξετάσουμε τις ΤΜΔ αλλά χρησιμοποιώντας μόνο συμπύκνωση. Χρησιμοποιούμε όμως και τον αλγόριθμο ENN που είναι αλγόριθμος επεξεργασίας και χρειάζεται για το επόμενο μέρος του πειράματος, αλλά σε αυτό το μέρος τον θεωρούμε ως έναν αλγόριθμο συμπύκνωσης. Εδώ δεν χρησιμοποιούμε καμία από τις μεθόδους δεικτοδότησης αλλά μόνο το brute force.

Στο τρίτο μέρος του πειράματος θα εξετάσουμε τις ΤΜΔ άλλα χρησιμοποιώντας πρώτα αλγορίθμους επεξεργασίας (ENN) για να φτιάξουμε σύνολα χωρίς θόρυβο και να τα εξετάσουμε με τις υπόλοιπες ΤΜΔ. Θα αναφερόμαστε σε αυτές ως ENN-CNN, ENN-IB2, ENN-RSP3 για να μην υπάρχει σύγχυση με τις ΤΜΔ του δευτέρου μέρους.

Και τα τρία μέρη του πειράματος είναι γραμμένα σε γλώσσα Python και εκτελέστηκαν στο Google Colab, το οποίο είναι δωρεάν διαδικτυακό περιβάλλον συγγραφής και εκτέλεσης κωδικά Python που παρέχεται από την Google με σκοπό την χρήση του από φοιτητές και ερευνητές πληροφορικής. Το Colab δημιουργεί μια εικονική μηχανή με επεξεργαστή Intel Xeon των 2.20GHz, 12GB μνήμη RAM και 100GB αποθηκευτικό χώρο που συνδέεται με το Google Drive, οπού και αποθηκεύετε ο κώδικας και τα αρχεία του πειράματος μας. Οι ΤΜΔ και ο διαχωριστής K-fold είναι υλοποιημένες σε C++ και εκτελέστηκαν σε έναν εξυπηρετητή του του Τμήματος Μηχανικής Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, ο οποίος έχει CPU με 12 πυρήνες 2.40 GHz, 16 GB RAM, λειτουργικό σύστημα Debian GNU/Linux 10 (buster).

4.3.1 Πειράματά χωρίς προ-επεξεργασίά

Πίνακας 4.2: Χρόνος Αναζήτησής σε δευτερόλεπτα για k = 5 χωρίς προ-επεξεργασία

A/A	Όνομά	Διαστάσεις	Μέγεθος	kd-tree	ball-tree	Brute Force
1	Banana	2	5300	0,19	0,21	0,83
2	Skin	3	245057	10,29	47,88	10003,03
3	Phoneme	5	5404	0,21	0,30	0,87
4	Shuttle	9	58000	3,29	12,65	54,08
5	MGT	10	19020	1,41	2,73	6,26
6	Poker	10	1025010	974,37	4625,16	14728,56
7	Wine	11	4898	0,34	0,43	0,69
8	Marketing	13	6876	0,49	0,85	1,14
9	PID	16	10992	0,96	1,57	2,37
10	LIR	16	20000	3,91	7,78	7,02
11	Ring	20	7200	2,26	1,67	1,33
12	KDD	36	141486	550,33	784,23	347,30

Στον Πίνακά 4.2, παρουσιάζονται τα αποτελέσματα για τον χρόνο αναζήτησης σε δευτερόλεπτα και στον Πίνακά 4.3 για την ακρίβειά κατηγοριοποίησης σε ποσοστό τις εκατό, για το πρώτο μέρος του πειράματος οπού δεν χρησιμοποιήθηκε καμία ΤΜΔ και τα σύνολα εξετάστηκαν με kd-tree, ball-tree, και brute force χρησιμοποιώντας την παράμετρο k=5. Στους πίνακές 4.4 και 4.5, παρουσιάζονται οι ίδιες μετρήσεις με τους προηγουμένους πίνακές άλλα έχοντας ως παράμετρο k το 1.

Και στις 2 περιπτώσεις το kd-tree είναι πιο γρήγορο και από το ball-tree και από το brute force για όσο οι διαστάσεις είναι λιγότερες από 20 και για 20 ή περισσότερες κερδίζει το brute force ενώ ταυτόχρονα το ball-tree είναι γρηγορότερο από το kd-tree αν και στο σύνολο δεδομένων KDD δεν ισχύει αυτό, άλλα το συγκεκριμένο σύνολο αποτελεί την εξαίρεση στον κανόνα.

Πίνακας 4.3: Ακρίβεια Κατηγοριοποίησης τις εκατό για k = 5 χωρίς προ-επεξεργασία

Α/Α	Όνομά	Διαστάσεις	Μέγεθος	kd-tree	ball-tree	Brute Force
1	Banana	2	5300	88,9	88,9	88,9
2	Skin	3	245057	99,95	99,95	99,95
3	Phoneme	5	5404	87,37	87,37	87,37
4	Shuttle	9	58000	99,87	99,87	99,87
5	MGT	10	19020	83,4	83,4	83.4
6	Poker	10	1025010	53,19	53,20	53,20
7	Wine	11	4898	55,63	55,63	55,63
8	Marketing	13	6876	29,35	29,43	29,21
9	PID	16	10992	93,23	93,23	93,23
10	LIR	16	20000	95,2	95,26	95,19
11	Ring	20	7200	68,79	68,79	68,79
12	KDD	36	141486	99,67	99,67	99,67

Πίνακας 4.4: Χρόνος Αναζήτησής σε δευτερόλεπτα για k = 1 χωρίς προ-επεξεργασία

Α/Α	Όνομά	Διαστάσεις	Μέγεθος	kd-tree	ball-tree	Brute Force
1	Banana	2	5300	0,15	0,17	0,61
2	Skin	3	245057	8,81	36,8	10003,03
3	Phoneme	5	5404	0,16	0,24	0,52
4	Shuttle	9	58000	2,85	13,00	26,30
5	MGT	10	19020	1,18	2,59	3,18
6	Poker	10	1025010	455,40	3048,48	6728,46
7	Wine	11	4898	0,26	0,39	0,47
8	Marketing	13	6876	0,39	0,75	0,78
9	PID	16	10992	0,69	1,47	1,33
10	LIR	16	20000	2,57	6,79	3,59
11	Ring	20	7200	2,26	1,70	0,86
12	KDD	36	141486	404,99	558,91	180,67

Πίνακας 4.5: Ακρίβεια Κατηγοριοποίησης τις εκατό για k = 1 χωρίς προ-επεξεργασία

Α/Α	Όνομά	Διαστάσεις	Μέγεθος	kd-tree	ball-tree	Brute Force
1	Banana	2	5300	86,75	86,75	86,75
2	Skin	3	245057	99,95	99,95	99,95
3	Phoneme	5	5404	89,69	89,69	89,69
4	Shuttle	9	58000	99,99	99,99	99,99
5	MGT	10	19020	80,90	80,90	80,90
6	Poker	10	1025010	50,19	50,19	50,18
7	Wine	11	4898	63,94	63,94	63,94
8	Marketing	13	6876	28,51	28,50	28,14
9	PID	16	10992	99,35	99,35	99,35
10	LIR	16	20000	95,8	95,8	95,74
11	Ring	20	7200	74,81	74,81	74,81
12	KDD	36	141486	99,7	99,7	99,7

4.3.2 Πειράματά μόνο με συμπύκνωσή

Στον πίνακα 4.6 παρουσιάζεται το ποσοστό μείωσης των συνόλων δεδομένων όταν χρησιμοποιούμε τις ΤΜΔ μόνο με συμπύκνωση. Έκτος του αλγορίθμου ENN που δημιουργεί ένα σύ-

νολο επεξεργασίας, βλέπουμε ότι η μείωση κυμαίνεται στο 13-99 % . Τα σύνολα που εχουν την μικρότερη μείωση παρουσιάζουν πολύ μεγάλη μείωση με τον αλγόριθμο ENN και αυτό δείχνει ότι περιέχουν πάρα πολύ θόρυβο. Π.χ το σύνολο Marketing με RSP3 παρουσιάζει 13 % μείωση ενώ με τον ENN 72 %. Υπάρχουν και σύνολα δεδομένων όπως το Shuttle ή το KDD στα οποία επιτυγχάνετε μείωση 99 % μέσω CNN και αυτό γίνετε γιατί δεν υπάρχει θόρυβος και δεν υπάρχουν πολλά αντικείμενα στα όρια απόφασης των κλάσεων.

Πίνακας 4.6: Ποσοστό μείωσης συνόλων δεδομένων τις εκατό

A/A	Όνομά	Μέγεθος	CNN	IB2	RSP3	ENN
1	Banana	5300	78	83	76	12
2	Skin	245057	99	99	99	—
3	Phoneme	5404	76	80	69	11
4	Shuttle	58000	99	99	94	0,10
5	MGT	19020	59	69	53	20
6	Poker	1025010	41	52	—	39
7	Wine	4898	36	46	31	52
8	Marketing	6876	19	25	13	72
9	PID	10992	95	96	89	0,70
10	LIR	20000	83	85	62	4
11	Ring	7200	72	79	56	29
12	KDD	141486	99	99	98	0,30
	M.O	—	71,3	76	67,3	21,8

Πίνακας 4.7: Χρόνος Αναζήτησης σε δευτερόλεπτα για k = 5 μόνο με συμπύκνωση

A/A	Όνομά	Διαστάσεις	CNN	IB2	RSP3	ENN
1	Banana	2	0,37	0,33	0,38	0,67
2	Skin	3	9,18	8,63	9,31	—
3	Phoneme	5	0,38	0,35	0,40	0,72
4	Shuttle	9	2,28	2,15	2,53	53,07
5	MGT	10	2,88	2,32	3,15	5,27
6	Poker	10	8919,73	6869,17	—	9261,54
7	Wine	11	0,55	0,53	0,57	0,47
8	Marketing	13	0,91	0,86	0,97	0,57
9	PID	16	0,64	0,59	0,70	2,37
10	LIR	16	1,71	1,63	2,82	6,85
11	Ring	20	0,63	0,57	0,76	0,98
12	KDD	36	7,12	6,58	8,13	304,12

Στον πίνακα 4.8 φαίνεται η ακρίβεια κατηγοριοποίησης και στον πίνακα 4.7 φαίνεται ο χρόνος αναζήτησης όταν εκτελούμε το brute force στα συμπυκνωμένα σύνολα και η παράμετρος είναι k=5. Αντίστοιχα στους πίνακες 4.10 και 4.9 φαίνονται οι ίδιες μετρήσεις όταν έχουμε ως παράμετρο k την τιμή 1.

Πίνακας 4.8: Ακρίβεια Κατηγοριοποίησης τις εκατό για k = 5 μόνο με συμπύκνωση

Α/Α	Όνομά	Διαστάσεις	CNN	IB2	RSP3	ENN
1	Banana	2	88,11	87,81	87,56	89,35
2	Skin	3	99,47	99,39	99,64	—
3	Phoneme	5	83,77	83,10	85,04	86,54
4	Shuttle	9	84,88	91,30	93,15	99,86
5	MGT	10	81,72	79,72	82,62	82,51
6	Poker	10	51,18	50,73	—	55,05
7	Wine	11	53,98	51,89	54,12	51,73
8	Marketing	13	28,84	27,46	28,27	30,49
9	PID	16	95,19	94,24	98,59	99,09
10	LIR	16	82,15	80,63	93,68	94,09
11	Ring	20	84,10	85,14	78,51	57,22
12	KDD	36	98,16	98,02	99,21	99,64

Πίνακας 4.9: Χρόνος Αναζήτησης σε δευτερόλεπτα για k = 1 μόνο με συμπύκνωση

Α/Α	Όνομά	Διαστάσεις	CNN	IB2	RSP3	ENN
1	Banana	2	0,30	0,28	0,38	0,50
2	Skin	3	7,90	7,66	8,20	—
3	Phoneme	5	0,31	0,30	0,35	0,56
4	Shuttle	9	1,92	1,90	2,05	26,14
5	MGT	10	1,82	1,50	1,90	2,63
6	Poker	10	3839,78	3786,22	—	4980,46
7	Wine	11	0,38	0,36	0,39	0,34
8	Marketing	13	0,69	0,70	0,74	0,45
9	PID	16	0,60	0,56	0,62	1,32
10	LIR	16	1,24	1,19	1,95	3,32
11	Ring	20	0,55	0,47	0,68	0,74
12	KDD	36	5,35	5,11	6,34	177,56

Πίνακας 4.10: Ακρίβεια Κατηγοριοποίησης τις εκατό για k = 1 μόνο με συμπύκνωση

Α/Α	Όνομά	Διαστάσεις	CNN	IB2	RSP3	ENN
1	Banana	2	85,71	83,49	83,83	88,56
2	Skin	3	99,94	99,90	99,76	—
3	Phoneme	5	87,78	85,38	86,75	87,78
4	Shuttle	9	86,33	86,79	93,16	99,89
5	MGT	10	76,15	74,41	78,15	82,17
6	Poker	10	46,96	45,95	—	53,15
7	Wine	11	59,18	56,86	58,57	53,22
8	Marketing	13	26,63	25,40	26,46	29,08
9	PID	16	98,68	98,04	99,05	99,29
10	LIR	16	92,52	91,59	95,40	94,93
11	Ring	20	82,68	81,58	81,71	62,17
12	KDD	36	99,63	99,46	99,59	99,67

Σκοπός μας βέβαια δεν είναι να συγκρίνουμε τις ΤΜΔ μεταξύ τους άλλα όπως βλέπουμε όταν χρησιμοποιείται ο αλγόριθμος IB2 έχουμε τους γρηγορότερους χρόνους αναζήτησής άλλα μερικές απώλειες στην ακρίβεια. Όταν χρησιμοποιείται ο αλγόριθμο RSP3 έχουμε συνήθως καλύτερο ποσοστό ακρίβειας άλλα η αναζήτηση παίρνει παραπάνω χρόνο. Και όταν χρησιμοποιείται ο αλγόριθμο CNN ήμαστε συνήθως μεταξύ των άλλων δυο αλγορίθμων και σε χρόνο αναζήτησης και σε ακρίβειά.Αυτά ισχύουν ανεξαρτήτως την παράμετρο k που θα χρησιμοποιήσουμε.

4.3.3 Πειράματα με επεξεργασία και συμπύκνωση

Πίνακας 4.11: Ποσοστό μείωσης συνόλων δεδομένων τις εκατό

A/A	Όνομά	Μέγεθος	ENN-CNN	ENN-IB2	ENN-RSP3
1	Banana	5300	93	94	91
2	Skin	245057	—	—	—
3	Phoneme	5404	91	92	86
4	Shuttle	58000	99	99	98
5	MGT	19020	90	91	84
6	Poker	1025010	80	83	—
7	Wine	4898	87	89	84
8	Marketing	6876	90	91	87
9	PID	10992	96	97	90
10	LIR	20000	87	88	66
11	Ring	7200	93	95	79
12	KDD	141486	99	99	99
	M.O	—	91,3	92,5	86,4

Πίνακας 4.12: Χρόνος Αναζήτησης σε δευτερόλεπτα για k = 5 με επεξεργασία και συμπύκνωση

A/A	Όνομά	Διαστάσεις	ENN-CNN	ENN-IB2	ENN-RSP3
1	Banana	2	0,27	0,27	0,28
2	Skin	3	—	—	—
3	Phoneme	5	0,30	0,29	0,32
4	Shuttle	9	2,16	2,07	2,39
5	MGT	10	1,25	1,16	1,49
6	Poker	10	3237,96	2639,80	—
7	Wine	11	0,30	0,29	0,32
8	Marketing	13	0,44	0,43	0,49
9	PID	16	0,59	0,58	0,68
10	LIR	16	1,58	1,48	2,60
11	Ring	20	0,41	0,40	0,55
12	KDD	36	5,64	5,58	6,59

Στον πίνακα 4.11 παρουσιάζεται το ποσοστό μείωσης των συνόλων δεδομένων όταν χρησιμοποιούνται οι ΤΜΔ με επεξεργασία και μετά με συμπύκνωση. Παρατηρείται ότι η μείωση κυμαίνεται μεταξύ 66-99 % . Όταν δεν χρησιμοποιείται επεξεργασία πριν από την συμπύκνωση η μείωση ήταν σε κάποιες περίπτωσης αρκετά μικρή. Όπως στην περίπτωση του συνόλου

Πίνακας 4.13: Ακρίβεια Κατηγοριοποίησης τις εκατό για k = 5 με επεξεργασία και συμπύκνωση

Α/Α	Όνομά	Διαστάσεις	ENN-CNN	ENN-IB2	ENN-RSP3
1	Banana	2	88,33	87,33	88,30
2	Skin	3	—	—	—
3	Phoneme	5	82,18	80,95	84,71
4	Shuttle	9	94,11	94,02	94,56
5	MGT	10	82,65	82,38	83,15
6	Poker	10	52,84	52,39	—
7	Wine	11	49,74	49,30	49,95
8	Marketing	13	29,39	28,90	28,37
9	PID	16	93,50	92,24	98,43
10	LIR	16	80,90	79,29	92,92
11	Ring	20	80,80	82,95	66,63
12	KDD	36	97,69	97,62	99,07

Πίνακας 4.14: Χρόνος Αναζήτησης σε δευτερόλεπτα για k = 1 με επεξεργασία και συμπύκνωση

Α/Α	Όνομά	Διαστάσεις	ENN-CNN	ENN-IB2	ENN-RSP3
1	Banana	2	0,26	0,25	0,27
2	Skin	3	—	—	—
3	Phoneme	5	0,26	0,26	0,30
4	Shuttle	9	1,86	1,83	2,01
5	MGT	10	1,01	0,98	1,21
6	Poker	10	1544,56	1414,08	—
7	Wine	11	0,26	0,25	0,27
8	Marketing	13	0,37	0,36	0,41
9	PID	16	0,58	0,55	0,60
10	LIR	16	1,13	1,06	1,79
11	Ring	20	0,37	0,36	0,49
12	KDD	36	4,48	4,42	5,53

Πίνακας 4.15: Ακρίβεια Κατηγοριοποίησης τις εκατό για k = 1 με επεξεργασία και συμπύκνωση

Α/Α	Όνομά	Διαστάσεις	ENN-CNN	ENN-IB2	ENN-RSP3
1	Banana	2	87,94	87,43	87,28
2	Skin	3	—	—	—
3	Phoneme	5	86,84	85,95	86,56
4	Shuttle	9	90,00	90,68	93,91
5	MGT	10	80,45	79,87	81,54
6	Poker	10	49,63	49,11	—
7	Wine	11	48,99	48,08	49,18
8	Marketing	13	27,74	26,92	26,14
9	PID	16	98,60	98,18	99,02
10	LIR	16	91,83	90,96	94,45
11	Ring	20	80,01	80,52	72,27
12	KDD	36	99,64	99,62	99,62

δεδομένων marketing που με RSP3 είχε μείωση 13 % ενώ με ENN-RSP3 φτάνει έως και 87 %. Σε άλλες περιπτώσεις οπού δεν υπήρχε πολύ θόρυβος, η αύξηση του ποσοστού μείωσης ήταν πολύ μικρότερη πως για παράδειγμα στο σύνολο Letter Image Recognition που με CNN είχαμε μείωση 83 % ενώ με ENN-CNN φτάνει το 87 %. Ενώ υπάρχουν και περιπτώσεις όπως το σύνολο KDD που από την αρχή έχουν τεράστια μείωση και δεν περιείχαν θόρυβο, και από το 99 % σαφώς και παρέμειναν στο 99 ΄%.

Στον πίνακα 4.12 παρουσιάζεται ο χρόνος αναζήτησης και στον πίνακα 4.13 η ακρίβεια του κατηγοριοποιητή k-NN για τιμή k=5. Ενώ αντίστοιχα στους πίνακες 4.14 και 4.15 παρουσιάζονται οι ίδιες μετρήσεις όμως για k=1.

Όπως παρατηρείται στους πίνακες οι γρηγορότεροι χρόνοι αναζήτησης παρουσιάζονται όταν χρησιμοποιείται η μέθοδος ENN-IB2 με την μέθοδο ENN-CNN να είναι ελάχιστα πιο αργή, ενώ η μέθοδος ENN-RSP3 είναι στην τελευταία θέση από άποψη ταχύτητας. Από άποψη ακρίβειας κατηγοριοποίησης, η μέθοδος ENN-IB2 παρουσιάζει σχεδόν πάντα χειρότερα αποτελέσματα από τις άλλες δυο μεθόδους. Ενώ υπάρχει "μάχη" ανάλογα με το σύνολο δεδομένων για το αν θα είναι πιο ακριβής η μέθοδος ENN-CNN ή η ENN-RSP3.

4.4 Σύγκριση αποτελεσμάτων

Σε αυτήν την υποενότητα θα συγκρίνουμε τα αποτελέσματα από τις μετρήσεις του πρώτου πειράματος για να αποφασίσουμε τελικά ποια μέθοδος είναι η καλύτερη ανάλογα με τη διάσταση. Θα ξεκινήσουμε από το σύνολο δεδομένων με τις λιγότερες διαστάσεις και θα εξετάζονται ένα ένα σε αύξουσα σειρά. Όταν αναφερόμαστε στο χρόνο εννοείται ο χρόνος αναζήτησης των εγγυτέρων γειτόνων και όταν αναφερόμαστε στην ακρίβεια εννοείται η ακρίβεια κατηγοριοποίησης στο σύνολο δεδομένων. Θα παρουσιάζεται πάντα η μείωση δεδομένων που προκαλούν οι ΤΜΔ γιατί είναι ένα μέτρο επιτυχίας των ΤΜΔ και όπως θα δούμε συσχετίζονται με τον χρόνο. Αρχικά αναλύονται τα αποτελέσματα για k=5 και θα επισημάνουμε αν υπάρχουν διαφορές όταν χρησιμοποιείται σαν παράμετρο το k=1.

Το πρώτο σύνολο δεδομένων που εξετάζεται είναι το Banana το οποίο έχει 2 διαστάσεις και 5300 αντικείμενα. Σύμφωνα με το πρώτο μέρος του πειράματος η πιο γρήγορη μέθοδος δεικτοδότησης είναι το kd-tree με χρόνο 0,19s ενώ το brute force χρειάζεται 0,83s και πετυχαίνουν ακρίβεια 88,90 %. Πηγαίνοντας στο δεύτερο μέρος, η μέθοδος CNN πετυχαίνει χρόνο 0,37s και ακρίβεια 88,11 % με 78 % μείωση του σύνολου δεδομένων, ενώ η μέθοδος IB2 με 83 % μείωση πετυχαίνει γρηγορότερο χρόνο άλλα χειρότερη ακρίβεια. Και οι δυο μέθοδοι εδώ είναι χειρότεροι από το kd-tree και σε χρόνο και σε ακρίβεια. Στο τρίτο μέρος η μέθοδος ENN-CNN πετυχαίνει ποσοστό μείωσης 93 % κρατώντας την ακρίβεια ψηλά στο 88,30 % και έχοντας χρόνο 0,27s. Όπως βλέπουμε στις 2 διαστάσεις η καλύτερη μέθοδος είναι με διάφορα το kd-tree, ακόμα και με 94 % μείωση οι ΤΜΔ είναι χειρότερες σε τόσο μικρο αριθμό διαστάσεων. Οι ΤΜΔ είναι πιο αργές και από το ball-tree το οποίο πετυχαίνει χρόνο 0,21s Αυτό είναι και το

αναμενόμενο μιας και οι μέθοδοι δεικτοδότησης λειτουργούν καλύτερα όσο λιγότερες είναι οι διαστάσεις. Όταν έχουμε ως παράμετρο k=1 έχουμε τα ίδια αποτελέσματα από άποψη χρόνου άλλα οι ΤΜΔ με επεξεργασία και συμπύκνωση παρουσιάζουν καλύτερη ακρίβεια από τις μεθόδους δεικτοδότησης.

Εξετάζοντας στο επόμενο σύνολο δεδομένων το οποίο είναι το Skin που έχει 3 διαστάσεις και 245057 αντικείμενα, σύμφωνα με το πρώτο μέρος του πειράματος παρατηρείται ότι η πιο γρήγορη μέθοδος δεικτοδότησης είναι και εδώ το kd-tree με χρόνο 10,29s ενώ το ball-tree κάνει 47,88s και το brute force 1003,03s και πετυχαίνουν ακρίβεια 99,95 %, εδώ φαίνεται ποσό χρήσιμες μπορεί να είναι οι μέθοδοι δεικτοδότησης αφού επιτυγχάνεται 100 φορές μικρότερος χρόνος από την εξαντλητική αναζήτηση. Στο δεύτερο μέρος, η μέθοδος CNN πετυχαίνει χρόνο 9,18s και ακρίβεια 99,47 % με 99,8 % μείωση του σύνολου δεδομένων, ενώ η μέθοδος IB2 πετυχαίνει σχεδόν παρόμοια αποτελέσματα ανταλλάσσοντας ένα μέρος της ακρίβειας για μικρότερους χρόνους. Και οι δυο μέθοδοι εδώ είναι αρκετά καλύτερες από το kd-tree και το ball-tree. Στο τρίτο μέρος παρουσιάστηκε πρόβλημα γιατί η μέθοδος ENN διέγραφε όλα τα αντικείμενα της 2ης κλάσης του συνόλου. Όταν έχουμε την παράμετρο k=1 παρατηρούμε τα ίδια αποτελέσματα άλλα δεν έχουμε καθόλου απώλειες στην ακρίβεια όταν χρησιμοποιούμε τις ΤΜΔ. Όπως βλέπουμε στο συγκεκριμένο σύνολο δεδομένων κερδίζουν οι ΤΜΔ άλλα όχι για μεγάλες διάφορες. Για αυτό ευθύνεται το ποσοστό μείωσης που είναι 99,8 % και έτσι μένει ένα πολύ μικρο μέρος από το αρχικό σύνολο εκπαίδευσης στο σύνολο συμπύκνωσης. Σε τόσες λίγες διαστάσεις οι μέθοδοι δεικτοδότησης είναι καλύτερες έκτος και αν οι ΤΜΔ μας εγγυηθούν ότι μπορούν να πετύχουν τεράστια μείωση, που πρακτικά είναι πάρα πολύ δύσκολο και συμβαίνει μόνο με ελάχιστα σύνολα δεδομένων.

Εστιάζοντας στο επόμενο σύνολο δεδομένων που είναι το Phoneme το οποίο έχει 5 διαστάσεις και 5404 αντικείμενα. Σύμφωνα με το πρώτο μέρος του πειράματος η καλύτερη μέθοδος δεικτοδότησης είναι και πάλι το kd-tree με 0,21s ακολουθεί το ball-tree με 0,30s και το brute force με 0,83s, πετυχαίνοντας 87,37 % ακρίβεια. Στο δεύτερο μέρος καλύτερη ακρίβεια παρουσιάζει η μέθοδος RSP3 με 85,04 % και χρόνο 0,40s έχοντας πετύχει 69 % μείωση δεδομένων, ενώ η πιο γρήγορη μέθοδος είναι η IB2 με 0,35s και μείωση 80 % άλλα έχει μεγάλες απώλειες σε ακρίβεια. Όταν έχουμε k=1 τότε καλύτερη μέθοδος είναι η CNN γιατί δεν παρουσιάζονται απώλειες σε ακρίβεια. Οι ΤΜΔ χωρίς επεξεργασία σε αυτό το σύνολο είναι χειρότερες και από τις δυο μεθόδους δεικτοδότησης. Στο τρίτο μέρος βλέπουμε ότι η καλύτερη ΤΜΔ είναι η ENN-RSP3 με μείωση 86 % ακρίβεια 84,71 % και χρόνο 0,32s, πάλι όμως είναι πιο αργή και με χειρότερη ακρίβεια και από τις δυο μεθόδους δεικτοδότησης. Η ENN-IB2 και ENN-CNN είναι λίγο γρηγορότερες από το ball-tree άλλα υστερούν σε ακρίβειας ενώ πετυχαίνουν 92 και 91 % μείωση αντίστοιχα. Για k=1 καλύτερη ΤΜΔ είναι η ENN-CNN άλλα δεν είναι καλύτερη από τις μεθόδους δεικτοδότησης. Όπως βλέπουμε ακόμα και η 92 % μείωση δεν είναι αρκετή για να καταστήσει τις ΤΜΔ πιο χρήσιμες σε τόσες λίγες διαστάσεις, θα έπρεπε να είχαμε μάλλον μια μείωση της τάξης του 99 % όπως και στο σύνολο Skin, όμως όπως είναι εμφανές τέτοια μείωση δεν είναι τόσο εύκολο να επιτευχθεί.

Αναβαίνοντας μερικές διαστάσεις εξετάζουμε το σύνολο δεδομένων Shuttle με 9 το οποίο περιέχει 58000 αντικείμενα. Ακόμα μια φορά το kd-tree είναι η πιο γρήγορη μέθοδος δεικτοδότησης με χρόνο 3,2s ακολουθεί το ball-tree με 12,65s και τελευταίο το brute force με 54s και 99,87% ακρίβεια. Στο δεύτερο μέρος του πειράματος οι μέθοδοι CNN και IB2 επιτυγχάνουν μείωση 99 % άλλα έχουμε τεράστιες απώλειες σε ακρίβεια, οπότε θα θεωρήσουμε καλύτερή ΤΜΔ την RSP3 με μείωση 94 %, χρόνο αναζήτησης 2,53s και ακρίβεια 93,15 %. Εδώ βλέπουμε ότι οι ΤΜΔ χωρίς επεξεργασία είναι ήδη πιο γρήγορες από τις μεθόδους δεικτοδότησης ακόμα και με 94 % μείωση. Στο τρίτο μέρος θεωρούμε καλύτερη μέθοδο την ENN-CNN που πετυχαίνει μείωσή 99 % με χρόνο αναζήτησης 2,16s και ακρίβεια 94,11 %. Για k=1 δεν παρατηρούμε σημαντικές διάφορές πέρα από ότι οι ΤΜΔ χάνουν παραπάνω σε ακρίβεια. Σε αυτό το σύνολο παρατηρούμε ότι οι ΤΜΔ δεν χρειάζεται πια να φτάσουν σε μείωση 99 % για να είναι πιο αποτελεσματικές από τις μεθόδους δεικτοδότησης. Αν μια ΤΜΔ καταφέρει να επιτύχει μείωση μεγαλύτερη από 90 % και δεν υπάρχουν μεγάλες απώλειες στην ακρίβεια τότε λογικά θα είναι και καλύτερη από τις μεθόδους δεικτοδότησης στις 9 διαστάσεις.

Επόμενο είναι το σύνολο δεδομένων Magic Gamma Telescope το οποίο έχει 10 διαστάσεις και 19020 αντικείμενα. Το kd-tree πάλι είναι η καλύτερη μέθοδος δεικτοδότησης με χρόνο αναζήτησης 1,41s ακολουθεί το ball-tree με 2,73s και το brute force με 6,26s και έχουν ακρίβεια 83,40%. Στο δεύτερο μέρος ο RSP3 είναι η καλύτερη ΤΜΔ αφού με 53 % μείωση πετυχαίνει χρόνο αναζήτησης 3,15s και ακρίβεια 82,62 %. Η μέθοδος IB2 είναι πιο γρήγορη με χρόνο 2,32s, που είναι πιο γρήγορος από το ball-tree άλλα όχι και από το kd-tree. Στο τρίτο μέρος καλύτερη μέθοδος είναι η ENN-IB2 που πετυχαίνει μείωση 90 % με ακρίβεια 82,38 % και έχει χρόνο 1,16s ο οποίος και είναι αρκετά πιο μικρός χρόνος από αυτόν του kd-tree. Για k=1 δεν παρατηρούμε κάτι διαφορετικό. Όπως βλέπουμε τώρα και με 90 % μείωση κερδίζουν οι ΤΜΔ άλλα όχι όταν είναι μόνο 84 % όπως η ENN-RSP3. Μια εκτίμηση για τις 10 διαστάσεις είναι ότι αν μια ΤΜΔ μπορεί να πετύχει μείωση μεγαλύτερη του 88-90 % τότε πιθανώς να είναι γρηγορότερη από τις μεθόδους δεικτοδότησης, έχοντας όμως μικρότερη ακρίβεια.

Το άλλο σύνολο με 10 διαστάσεις είναι το Poker το οποίο είναι και το μεγαλύτερο σύνολο με 1025010 αντικείμενα. Εδώ ο χρόνος αναζήτηση του brute force είναι 14728s που είναι περίπου 4 ώρες με ακρίβεια 53,20 %, αυτός ο χρόνος είναι απαγορευτικός και καθιστά την εξαντλητική αναζήτηση άχρηστη. Ο χρόνος του kd-tree είναι 974s ενώ του ball-tree 4625s δηλαδή λίγο παραπάνω από 1 ώρα, καλύτερος χρόνος από το brute force άλλα πάλι απαγορευτικός. Στο δεύτερο μέρος η καλύτερη μέθοδος είναι η IB2 με μείωση 52 %, χρόνο αναζήτησης 6869s και ακρίβεια 50,73 %, που όμως είναι χειρότερη και από τις δυο μεθόδους δεικτοδότησης. Αν χρησιμοποιήσουμε επεξεργασία πριν την συμπύκνωση καλύτερη μέθοδος είναι η ENN-IB2 η οποία πετυχαίνει 83 % μείωση, 52,39 % ακρίβειά και 2639s χρόνο αναζήτησης. Εδώ το kd-tree αποδεικνύετε πως είναι η καλύτερη μέθοδος, ίσως αν οι ΤΜΔ πετύχαιναν τουλάχιστον 90 % μείωση τότε να ήταν αυτές καλύτερες στις 10 διαστάσεις, γιατί κάτι παρόμοιο συνέβη με το προηγούμενο σύνολο που είναι και αυτό των 10 διαστάσεων, άλλα δεν συμβαίνει στο συγκεκριμένο σύνολο δεδομένων. Για k=1 δεν παρατηρούμε κάτι διαφορετικό. Δυστυχώς δεν μπορέσαμε να

χρησιμοποιήσουμε εδώ τον RSP3, καθώς η κατασκευή του συνόλου συμπύκνωσης απαιτούσε υπερβολικό χρόνο προ-επεξεργασίας.

Εξετάζουμε επόμενο σύνολο δεδομένων το οποίο είναι το Winequality white με 11 διαστάσεις και 4898 αντικείμενά. Εδώ η καλύτερή μέθοδος δεικτοδότησης είναι το kd-tree με 0,34s χρόνο αναζήτησης ενώ το brute force κάνει χρόνο 0,69s και επιτυγχάνουν 55,63 % ακρίβεια. Στο δεύτερο μέρος καλύτερη ΤΜΔ είναι η CNN με ακρίβειά 51,89 % και χρόνο αναζήτησης 0,55s έχοντας πετύχει μείωση 36 %, η μείωση είναι πολύ μικρή και έτσι κερδίζουν προς το παρών και οι δυο μέθοδοι δεικτοδότησης. Εδώ παρατηρούμε ότι για k=1 όλες οι ΤΜΔ είναι πιο γρήγορες άλλα χειρότερες σε ακρίβεια από το ball-tree. Στο τρίτο μέρος ακόμα και με 84 % μείωση οι ΤΜΔ είναι πιο γρήγορες, αλλά υστερούν σε ακρίβεια. Η μέθοδος ENN-RSP3 πετυχαίνει ακρίβεια 49,95 % δηλαδή περίπου 6 % απώλειά και χρόνο 0,32s. Οι άλλες ΤΜΔ είναι πιο γρήγορες άλλα έχουν ακόμα μεγαλύτερες απώλειες σε ακρίβεια. Στο συγκεκριμένο σύνολο λόγο της απώλειας σε ακρίβεια είναι καλύτερα να χρησιμοποιήσουμε το kd-tree αντί για τις ΤΜΔ αλλά όπως βλέπουμε σε αντίθεση τα σύνολα των 10 ή και λιγότερων διαστάσεων, δεν χρειάζεστε πια τουλάχιστον 90 % μείωση για να κερδίζουν οι ΤΜΔ σε ταχύτητα.

Το επόμενο σύνολο δεδομένων είναι το Marketing με 13 διαστάσεις και 6876 αντικείμενα. Εδώ παρατηρούμε πάρα πολύ μικρή ακρίβεια και αυτό συμβαίνει γιατί το σύνολο προέρχεται από ονομαστικά χαρακτηριστικά που μετατράπηκαν σε αριθμητικά. Καλύτερη μέθοδος και εδώ είναι το kd-tree με χρόνο αναζήτησης 0,49s και ακρίβεια 28,51 %. Εδώ η ακρίβεια των μεθόδων δεικτοδότησης είναι διαφορετική από αυτήν του brute force, επίσης βλέπουμε ότι το ball-tree έχει την ίδια ταχύτητα με το brute force όταν η παράμετρος k=1. Στο δεύτερο μέρος καλύτερη ΤΜΔ είναι η ENN, κάτι που βλέπουμε πρώτη φορά λόγο του θορύβου που υπάρχει στο σύνολο, με μείωση 72 %, χρόνο αναζήτησης 0,57s και ακρίβεια 30,49 %, ενώ είναι καλύτερη από το ball-tree. Οι υπόλοιπες ΤΜΔ πετυχαίνουν μέχρι 25 % μείωση. Όταν κάνουμε πρώτα επεξεργασία και μετά συμπύκνωση το ποσοστό μείωσης είναι πια σε ικανοποιητικά επίπεδα. Καλύτερη μέθοδος εδώ είναι η ENN-CNN με ποσοστό μείωσης 90 %, χρόνο αναζήτησης 0,44s και ακρίβεια 29,39 %, και έτσι πετυχαίνει και μεγαλύτερη ακρίβεια και μικρότερο χρόνο από το brute force. Σε αυτό το σύνολο οι ΤΜΔ χρειάζεται να φτάσουν τουλάχιστον 87 % για να είναι το ίδιο αποτελεσματικές με τις μεθόδους δεικτοδότησης.

Εστιάζοντας στο επόμενο σύνολο δεδομένων το οποίο είναι το Pendigits με 16 διαστάσεις και 10992 αντικείμενα. Η καλύτερη μέθοδος δεικτοδότησης είναι ακόμα το kd-tree, αν και έχουμε ανέβει σε διαστάσεις, με χρόνο αναζήτησης 0,96s και ακρίβεια 99,23 %. Στο δεύτερο μέρος καλύτερη ΤΜΔ είναι η RSP3 που με μείωση 89 % πετυχαίνει χρόνο αναζήτησης 0,70s και κρατάει την ακρίβεια ψηλά στο 98,56 %, αυτός ο χρόνος είναι πολύ καλύτερος από αυτούς των μεθόδων δεικτοδότησης. Οι άλλες ΤΜΔ πετυχαίνουν αρκετά γρηγορότερους χρόνους άλλα έχουν πολύ κακές επιδόσεις στην ακρίβεια. Στο τρίτο μέρος καλύτερη ΤΜΔ είναι η ENN-RPS3 με μείωση 90 %, χρόνο αναζήτησης 0,68s και ακρίβεια 98,43 %. Όταν όμως χρησιμοποιούμε το k=1 τότε όλες οι ΤΜΔ παρουσιάζουν πολύ καλύτερα αποτελέσματα από τις μεθόδους δεικτοδότησης,

ενώ δεν έχουμε σημαντικές απώλειες στην ακρίβεια και επιπλέον ο χρόνος του brute force είναι καλύτερος από τον αντίστοιχο του ball-tree. Εδώ είναι πιθανό ότι ακόμα και με ποσοστό μείωσης 80 % οι ΤΜΔ να είναι καλύτερες από τις μεθόδους δεικτοδότησης.

Στον ίδιο αριθμό διαστάσεων αλλά με 20000 αντικείμενα έχουμε το επόμενο σύνολο, το οποίο είναι το Letter Image Recognition. Εδώ καλύτερη μέθοδος δεικτοδότησης είναι το kd-tree με χρόνο αναζήτησης 3,91s και ακρίβεια 95,20 %. Εδώ παρατηρούμε ότι το ball-tree γίνετε αρκετά αναποτελεσματικό και έτσι το brute force είναι πιο γρήγορο χωρίς καμία ΤΜΔ. Στο δεύτερο μέρος καλύτερη ΤΜΔ είναι η RSP3 με μείωση 62 %, χρόνο αναζήτησης 2,87s και ακρίβεια 93,68 %. Εδώ ο αλγόριθμος RSP3 ακόμα και με ένα μέτριο ποσοστό μείωσης επιτυγχάνει πολύ καλύτερα αποτελέσματα από τις μεθόδους δεικτοδότησης. Οι άλλες ΤΜΔ πετυχαίνουν πολύ πιο γρήγορους χρόνους αλλά έχουν τεράστιες απώλειες στην ακρίβεια. Στο τρίτο μέρος καλύτερη ΤΜΔ είναι η ENN-RPS3 με μείωση 66 %, χρόνο αναζήτησης 2,60s και ακρίβεια 92,92 %. Ενώ οι άλλες ΤΜΔ έχουν πάλι πάρα πολύ κακά αποτελέσματα στην ακρίβεια. Όταν έχουμε για παράμετρο το k=1 τότε όλες οι ΤΜΔ παρουσιάζουν πολύ καλύτερα αποτελέσματα. Σε αυτό το σύνολο είναι πολύ πιθανών ότι ακόμα και μόνο με 50 % μείωση οι ΤΜΔ θα έχουν καλύτερα αποτελέσματα από τις μεθόδους δεικτοδότησης.

Αν και τα δυο παραπάνω σύνολα έχουν τις ίδιες διαστάσεις, απαιτείται να επιτευχθεί διαφορετικό ποσοστό μείωσης ώστε οι ΤΜΔ να είναι αποτελεσματικότερες από τις μεθόδους δεικτοδότησης. Αυτό είναι φυσιολογικό μιας και είδαμε ότι η αποτελεσματικότητα των μεθόδων δεικτοδότησης δεν επηρεάζεται μόνο από των αριθμό των διαστάσεων άλλα και από την δομή των δεδομένων ενός συνόλου.

Εξετάζοντας το σύνολο Ringnorm το οποίο αποτελείται από 20 διαστάσεις και 7200 αντικείμενα, σύμφωνα με το scikit-learn έχουμε περάσει το όριο των διαστάσεων που είναι αποτελεσματικό το kd-tree. Και όπως βλέπουμε αυτό επαληθεύεται και με τα αποτελέσματα μας. Εδώ η καλύτερη μέθοδος δεικτοδότησης είναι το ball-tree με χρόνο αναζήτησης 1,67s και ακρίβεια 68,79 %. Όμως και οι δυο μέθοδοι δεικτοδότησης είναι χειρότεροι από το brute force. Εδώ δεν χρειάζεται καν να συγκρίνουμε τις ΤΜΔ με τις μεθόδους δεικτοδότησης γιατί υπάρχει ξεκάθαρος νικητής. Άλλα παρατηρούμε σε αυτό το σύνολο ότι όταν χρησιμοποιούμε τον IB2 έχουμε τεράστια αύξηση στην απόδοση, επιτυγχάνοντας μείωση 79 %, χρόνο αναζήτησης 0,57s και ακρίβεια 85,14 % που είναι περίπου 15 % αύξηση. Όταν έχουμε k=1 δεν παρατηρούμε κάτι διαφορετικό.

Το τελευταίο σύνολο δεδομένων είναι το KDD με 36 διαστάσεις και 141486 αντικείμενα. Σε αυτό το σύνολο αν και έχουμε τόσες πολλές διαστάσεις το kd-tree είναι καλύτερο από το ball-tree άλλα αυτό είναι λόγο ιδιαιτερότητας του συνόλου. Ο χρόνος αναζήτησης του kd-tree είναι 550s ενώ του ball-tree 784s. Φυσικά και οι δυο μέθοδοι δεικτοδότησης είναι πολύ πιο αργές από το brute force που έχει χρόνο 347s. Ούτε εδώ έχει νόημα να συγκρίνουμε τις μεθόδους αλλά βλέπουμε ότι με την μέθοδο ENN-RSP3 που πετυχαίνει μείωση 99 % χρειαζόμαστε μόνο

6,59s για την αναζήτηση ενώ κρατάμε την ακρίβεια σε υψηλά ποσοστά.

4.5 Πως η κατάρα των διαστάσεων επηρεάζει τις μεθόδους δεικτοδότησης

Σε αυτήν την υποενότητα θα δούμε το πως η κατάρα των διαστάσεων μπορεί να επηρεάσει τις μεθόδους δεικτοδότησης. Θα χρησιμοποιήσουμε το πολύ γνωστό λογισμικό εξόρυξης δεδομένων "Weka" [16]. Και μέσω αυτού θα εξετάσουμε τα δυο μεγαλύτερα σύνολα δεδομένων μας, δηλαδή το KDD και το Ringnorm, χρησιμοποιώντας επιλογή χαρακτηριστικών (Feature selection). Η επιλογή χαρακτηριστικών θα γίνει μέσω του κριτηρίου Gain Ratio [65], το οποίο θα μας εμφανίσει μια λίστα με την κατάταξη των χαρακτηριστικών που πληρούν περισσότερο το κριτήριο. Θα εξετάζουμε τις μεθόδους δεικτοδότησης και το brute force με ένα αυξανόμενο υποσύνολο των χαρακτηριστικών, μέχρι να εξετάσουμε το πλήρες σύνολο.

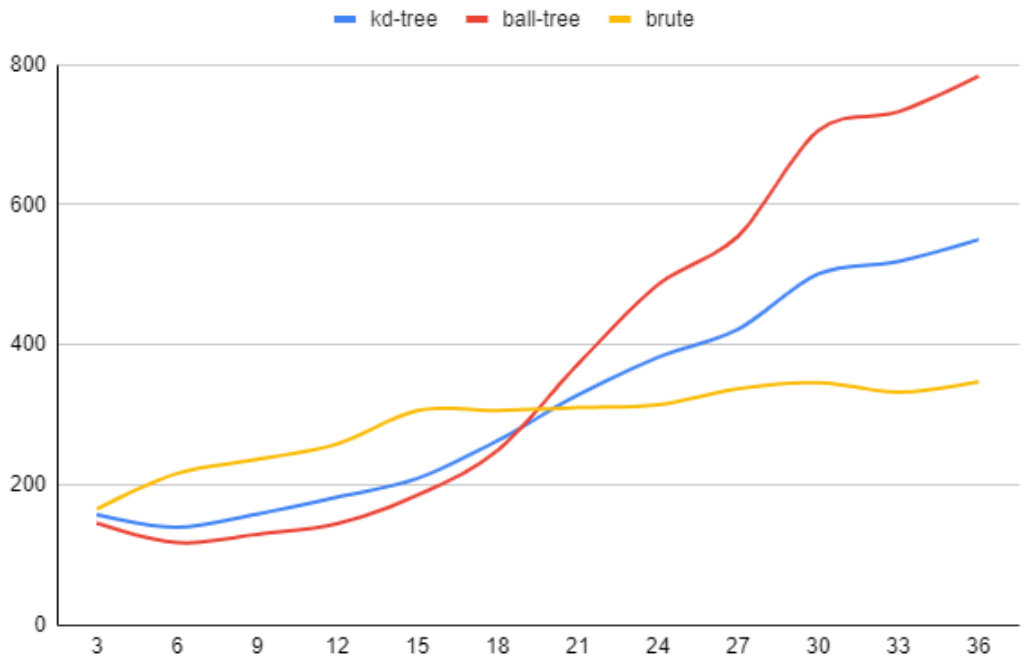
Το Waikato Environment for Knowledge Analysis (Weka), αναπτύχθηκε από Πανεπιστήμιο του Γουαϊκάτο, Νέα Ζηλανδία. Είναι ελεύθερο λογισμικό υπό την άδεια χρήσης GNU General Public License, και το λογισμικό που συνοδεύει το βιβλίο "Data Mining: Practical Machine Learning Tools and Techniques". Το Weka περιέχει μια συλλογή από εργαλεία οπτικοποίησης και αλγορίθμων ανάλυσης δεδομένων, μαζί με γραφικές διεπαφές για εύκολη πρόσβαση των χρηστών στις λειτουργίες αυτές [16]. Η έκδοση με βάση την Java (Weka 3), για την οποία η ανάπτυξη ξεκίνησε το 1997, χρησιμοποιείται τώρα σε πολλούς διαφορετικούς τομείς, κυρίως για εκπαιδευτικούς και ερευνητικούς σκοπούς .Το Weka υποστηρίζει διάφορες βασικές διεργασίες εξόρυξης δεδομένων όπως η προ-επεξεργασία δεδομένων, η συσταδοποίηση, η κατηγοριοποίηση, η παλινδρόμηση, ηοπτικοποίηση, οι κανόνες συσχέτισης και επιλογή χαρακτηριστικών.

Πίνακας 4.16: Χρόνος Αναζήτησης στο Σύνολο KDD μέσω Weka Gain Ratio σε δευτερόλεπτα

Διασταση	Kd-tree	Ball-tree	Brute
3	157	145	165
6	139	117	216
9	158	129	236
12	182	144	258
15	209	185	306
18	263	249	306
21	328	372	310
24	382	486	314
27	422	555	337
30	501	706	346
33	519	733	332
36	550	784	347

Πίνακας 4.17: Ακρίβεια Κατηγοριοποίησης στο Σύνολο KDD μέσω Weka Gain Ratio τις εκατό

Διασταση	Kd-tree	Ball-tree	Brute
3	62,15	62,15	59,08
6	91,38	91,38	91,38
9	97,95	97,95	97,97
12	91,72	91,72	98,07
15	99,22	99,22	99,22
18	99,39	99,39	99,39
21	98,78	98,78	98,65
24	99,50	99,50	99,50
27	99,63	99,63	99,63
30	99,65	99,65	99,65
33	99,66	99,66	99,66
36	99,67	99,67	99,67



Σχήμα 4.1: Γράφημα χρόνων αναζήτησης των μεθόδων δεικτοδότησης στο σύνολο KDD

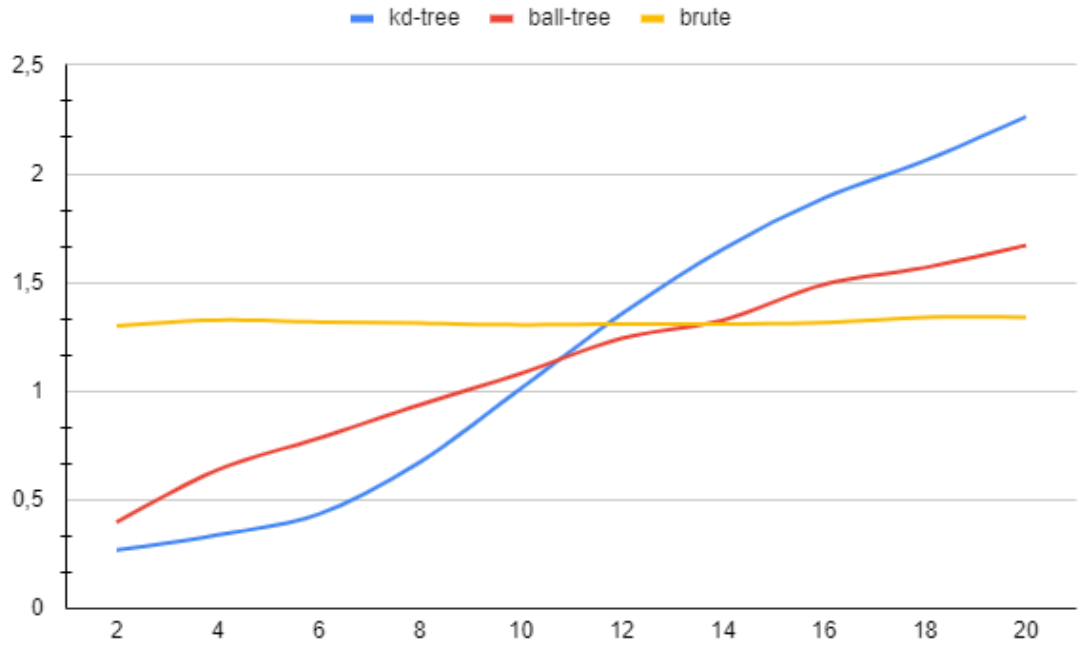
Στους πίνακες 4.16 και 4.17 βλέπουμε να αποτελέσματα από το πείραμα της επιλογής χαρακτηριστικών στο σύνολο δεδομένων KDD για τον χρόνο και την ακρίβεια αντίστοιχα. Το σχήμα 4.1 παρουσιάζει το γράφημα από τον πίνακα 4.16. Στον άξονα x έχουμε τις διαστάσεις ξεκινώντας από τις 3 και ανεβαίνοντας με βήμα 3 μέχρι και να φτάσουμε στις 36, οπού και είναι όλες οι διαστάσεις του συνόλου, ενώ στον άξονα y έχουμε τον χρόνο αναζήτησης σε δευτερόλεπτα. Όπως βλέπουμε όταν είμαστε σε λιγότερες από 6 διαστάσεις τότε η ακρίβεια είναι στο 62 % και ο χρόνος είναι μεγαλύτερος από τον χρόνο των 6 διαστάσεων για τις μεθόδους δεικτοδότησης. Αυτό συμβαίνει λόγο της εγγενής διάστασης που είδαμε, δεν υπάρχουν δηλαδή επαρκής

Πίνακας 4.18: Χρόνος Αναζήτησης στο Σύνολο Ringnorm μέσω Weka Gain Ratio σε δευτερόλεπτα

Διασταση	Kd-tree	Ball-tree	Brute
2	0,26	0,39	1,30
4	0,33	0,63	1,32
6	0,43	0,78	1,31
8	0,67	0,93	1,31
10	1,01	1,08	1,30
12	1,35	1,24	1,30
14	1,65	1,32	1,30
16	1,88	1,49	1,31
18	2,06	1,56	1,33
20	2,26	1,67	1,33

Πίνακας 4.19: Ακρίβεια Κατηγοριοποίησης στο Σύνολο Ringnorm μέσω Weka Gain Ratio τις εκατό

Διασταση	Kd-tree	Ball-tree	Brute
2	69,62	69,62	69,62
4	78,27	78,27	78,27
6	82,45	82,45	82,45
8	83,93	83,93	83,93
10	83,22	83,22	83,22
12	80,83	80,83	80,83
14	77,85	77,85	77,85
16	75,04	75,04	75,04
18	71,68	71,68	71,68
20	68,79	68,79	68,79



Σχήμα 4.2: Γράφημα χρόνων αναζήτησης των μεθόδων δεικτοδότησης στο σύνολο Ringnorm

πληροφορίες για την αναπαράσταση των δεδομένων. Με το που φτάσουμε τις 6 διαστάσεις η ακρίβεια φτάνει στο 91 % από εκεί και πέρα η ακρίβεια αυξάνεται σταδιακά με κάθε 3 διαστάσεις που προσθέτουμε μέχρι να φτάσουμε στο 99,67 % στις 36 διαστάσεις, παρατηρούμε όμως ότι στις 12, μόνο για τις μεθόδους δεικτοδότησης, και στις 21 διαστάσεις έχουμε μια πτώση στην ακρίβεια σε σχέση με το προηγούμενο βήμα

Ας εξετάσουμε τώρα τα αποτελέσματα του χρόνου αναζήτησης. Στο σχήμα 4.1 η μπλε γραμμή απεικονίζει τον χρόνο του kd-tree, η κόκκινη τον χρόνο του ball-tree και τέλος η κίτρινη τον χρόνο του brute force. Όπως είπαμε λόγο της εγγενής διάστασης δεν έχουμε αρκετές πληροφορίες και έτσι τα δέντρα που σχηματίζονται από τις μεθόδους δεικτοδότησης παρουσιάζουν προβλήματα κατά την αναζήτηση. Για αυτό και ο χρόνος αναζήτησης στο ball-tree για 3 διάστασης είναι μεγαλύτερος από αυτόν για 12 διαστάσεις. Για το kd-tree παρατηρούμε ότι ο χρόνος ανεβαίνει με μικρά βήματα μέχρι τις 15 διαστάσεις. Από εκεί και μετά φτάνει σε ένα σημείο που κάνει τεράστια βήματα με κάθε τρεις διαστάσεις που προσθέτουμε, π.χ για να πάμε από τις 18 στις 21 διαστάσεις χρειαζόμαστε ένα λεπτό παραπάνω. Το ball-tree και αυτό όπως και το kd-tree ανεβαίνει με μικρά βήματα μέχρι τις 15 διαστάσεις άλλα όπως βλέπουμε από τις 18 διαστάσεις και πάνω ανεβαίνει με τεράστιους ρυθμούς, πολύ μεγαλύτερους από αυτούς του kd-tree. Έτσι καταλαβαίνουμε ότι η απόδοση των μεθόδων δεικτοδότησης πέφτει σε μεγάλο βαθμό, λόγο της κατάρας των διαστάσεων, όταν έχουμε παραπάνω από 15 διαστάσεις. Το brute force βλέπουμε ότι ανεβαίνει σιγά σιγά μέχρι που φαίνεται να σταθεροποιείται ανάμεσα στα 330-350s. Εδώ λόγο της δομής του συνόλου KDD βλέπουμε ότι το ball-tree είναι καλύτερο από το kd-tree σε μικρό αριθμό διαστάσεων άλλα χειρότερο όταν αυξάνονται οι διαστάσεις. Θεωρητικά θα έπρεπε να συμβαίνει το τελείως αντίθετο. Άλλα όπως είπαμε η απόδοση των μεθόδων δεικτο-

δότησης έχει να κάνει και σε μεγάλο βαθμό από την δομή του συνόλου δεδομένων, ωστόσο το γράφημά παρουσιάζει πολύ καλά το πως επηρεάζονται οι μέθοδοι δεικτοδότησης από την κατάρα των διαστάσεων.

Στη συνέχεια εξετάζουμε το σύνολο δεδομένων Ringnorm. Στους πίνακες 4.18 και 4.19 βλέπουμε τα αποτελέσματα από το πείραμα της επιλογής χαρακτηριστικών στο συγκεκριμένο σύνολο δεδομένων για τον χρόνο και την ακρίβεια. Όπως βλέπουμε εδώ όταν υπάρχουν μόνο 2 διαστάσεις δεν έχουμε αρκετές πληροφορίες για σωστή αναπαράσταση των δεδομένων και έτσι έχουμε μόνο 69 % ακρίβεια σε σχέση με το 78 % που έχουμε στις 4 διαστάσεις. Η ακρίβεια ανεβαίνει από τις 2 μέχρι τις 10 διαστάσεις και πέφτει από τις 12 μέχρι τις 20 φτάνοντας σε ακρίβειά χειρότερη από αυτήν των 2 διαστάσεων. Αυτό γίνεται γιατί μετά τις 10 διαστάσεις επιβαρύνουμε με περιττές πληροφορίες για κάθε επιπλέον χαρακτηριστικό που προσθέτουμε. Η επιλογή χαρακτηριστικών θεωρείται ως μια μέθοδος μείωσης διαστάσεων με σκοπό την αποφυγή της κατάρας των διαστάσεων. Σε αυτό το σύνολο φαίνεται ποσό χρήσιμη μπορεί να γίνει η επιλογή χαρακτηριστικών ή και κάποια άλλη μέθοδος μείωσης διαστάσεων [26].

Θα εξετάσουμε τα αποτελέσματά του χρόνου σύμφωνα με τον πίνακα 4.18 και το σχήμα 4.2. Στο 4.2 σχήμα η μπλε γραμμή απεικονίζει τον χρόνο του kd-tree, η κόκκινη τον χρόνο του ball-tree και τέλος η κίτρινη τον χρόνο του brute force. Στο kd-tree βλέπουμε μια πολύ μικρή αύξηση μέχρι τις 6 διαστάσεις και από τις 6 μέχρι και τις 20 έχουμε μια μεγαλύτερη αύξηση, ανάμεσα στα 20-30s ανά κάθε 2 διαστάσεις που προσθέτουμε. Στο ball-tree βλέπουμε το αντίθετο, ξεκινάμε από μεγαλύτερο χρόνο για τις 2 διαστάσεις και μέχρι τις 6 έχουμε μεγάλη αύξηση για κάθε 2 επιπλέον διαστάσεις. Από τις 6 μέχρι και τις 20 διαστάσεις αυτή η αύξηση μειώνεται, σε περίπου 10s ανά κάθε 2 διαστάσεις που προσθέτουμε. Για το brute force παρατηρούμε ότι ο χρόνος αναζήτησης είναι σταθερά ανάμεσα σε 1,30-1,33s ανεξάρτητα από την διάσταση στην οποία βρισκόμαστε. Βλέπουμε λοιπόν ότι η καλύτερη μέθοδος είναι το kd-tree από τις 2 έως τις 10 διαστάσεις. Μόνο για τις 12 διαστάσεις καλύτερη μέθοδος είναι το ball-tree. Και από τις 14 έως τις 20 διαστάσεις καλύτερη μέθοδος γίνεται το brute force. Εδώ οι μέθοδοί δεικτοδότησης γίνονται αναποτελεσματικοί ακόμα και από τις 14 διαστάσεις, που είναι απολυτά φυσιολογικό.

Σε αυτά τα δυο σύνολα βλέπουμε μια γενική ιδέα για το πως επηρεάζονται οι μέθοδοι δεικτοδότησης από την κατάρα των διαστάσεων. Όπως καταλαβαίνουμε η απόδοση των μεθόδων δεικτοδότησης μπορεί να πέσει ακόμα και όταν βρισκόμαστε σε λιγότερες από 20 διαστάσεις. Για αυτό τον λόγο η παράμετρος “auto” στον κατηγοριοποιητή K-NN επιλέγει το brute force για $D > 15$, ώστε να αποφύγει τελείως τις περιπτώσεις που οι μέθοδοι δεικτοδότησης πέφτουν σε απόδοση από τις 15 διαστάσεις ακόμα και αν αυτό δεν γίνει τελικά.

4.6 Επίλογος

Σε αυτό το κεφάλαιο αρχικά παρουσιάστηκαν τα σύνολα δεδομένων που χρησιμοποιήθηκαν στην πειραματική μελέτη της παρούσας εργασίας. Στη συνέχεια το πείραμα χωρίστηκε σε τρία

μέρη και εξηγήθηκαν οι παράμετροι του πειράματος. Όλα τα αποτελέσματα καταγράφηκαν σε πίνακές και συγκρίθηκαν αναλυτικά για το κάθε σύνολο δεδομένων. Έπειτα πραγματοποιήθηκε και το δεύτερο πείραμα.

Έτσι φαίνεται εύκολα ένα αρχικό συμπέρασμα, όσο ανεβαίνουν οι διαστάσεις τόσο πέφτει η απόδοση των μεθόδων δεικτοδότησης και έτσι γίνεται πιο αποτελεσματική η χρήση κάποιας ΤΜΔ. Επιπλέον πέρα από τις διαστάσεις η ταχύτητα των μεθόδων δεικτοδότησης επηρεάζεται και από την δομή του συνόλου. Η ταχύτητα των ΤΜΔ εξαρτάται άμεσα από το ποσοστό μείωσης του συνόλου. Αν οι ΤΜΔ καταφέρουν να επιτύχουν τεράστιο ποσοστό μείωσης, μπορεί να είναι πιο αποδοτικές από τις μεθόδους δεικτοδότησης ακόμα και σε λίγες διαστάσεις (π.χ 5).

Κεφάλαιο 5ο: Συμπεράσματα και μελλοντική έρευνα

Στόχος αυτής της πτυχιακής εργασίας ήταν η σύγκριση των μεθόδων δεικτοδότησης και των ΤΜΔ για την κατηγοριοποίηση εγγύτερων γειτόνων. Έκτος αυτού έγινε ένα επιπλέον πείραμα για το πως επηρεάζονται οι μέθοδοι δεικτοδότησης από την διάσταση. Παρακάτω παρουσιάζονται τα τελικά συμπεράσματα της πειραματικής ανάλυσης, χωρισμένα σε εύρη διαστάσεων (2-6,7-10,11-14,15-19,20+).

Όταν η διάσταση είναι μεταξύ 2 και 6 τότε ξεκάθαρα οι μέθοδοι δεικτοδότησης είναι καλύτερες, πιο συγκεκριμένα το kd-tree, από τις ΤΜΔ έκτος αν καταφέρουν να επιτύχουν μείωση μεγαλύτερη του 99 %. Αυτό βέβαια δεν είναι κάτι συνηθισμένο γιατί όπως είδαμε και στις πειραματικές μετρήσεις, μόνο λίγα σύνολα δεδομένων κατάφεραν να έχουν τέτοιο ποσοστό μείωσης.

Όταν η διάσταση είναι μεταξύ 7 και 10 τότε η διάκριση της καλύτερης μεθόδου γίνετε μια δύσκολη διαδικασία. Αν και βλέπουμε ότι με ποσοστό μείωσης 90 % οι ΤΜΔ μπορεί να είναι πιο γρήγορες από τις μεθόδους δεικτοδότησης, δεν είναι σίγουρο ότι θα φτάσουμε αυτόν τον αριθμό. Ο μέσος ορός της μείωσης με τη μέθοδο ENN-CNN είναι 91 %, αλλά αυτός ο μέσος ορός ανεβαίνει λόγο των συνόλων που μπορούν να μειωθούν μέχρι και 99 %. Επιπλέον μπορεί να έχουμε μειωμένη ακρίβεια όταν χρησιμοποιούμε κάποια ΤΜΔ. Άρα το συμπέρασμα εδώ είναι ότι οι μέθοδοι δεικτοδότησης είναι σε γενικές γραμμές πιο αποτελεσματικές και ασφαλής σε αυτές τις διαστάσεις, γιατί ακόμα και αν έχουν την ίδια ταχύτητα με τις ΤΜΔ θα παρουσιάζουν μεγαλύτερη ακρίβεια. Αλλά δεν μπορούμε να είμαστε σίγουροι ότι οι ΤΜΔ θα έχουν χειρότερα αποτελέσματα εκτός και αν τις εφαρμόσουμε πρώτα στο σύνολο δεδομένων.

Όταν η διάσταση είναι μεταξύ 11 και 14 τότε ανάλογα με την δομή του κάθε συνόλου και το πόσες ακριβώς διαστάσεις έχει, χρειαζόμαστε περίπου 80 % μείωση για να είναι πιο αποτελεσματικές οι ΤΜΔ. Είναι εμφανές ότι όσο πλησιάζουμε στις 15 διαστάσεις τόσο μικρότερο ποσοστό μείωσης χρειαζόμαστε. Επειδή ο μέσος ορός του ENN-CNN είπαμε ότι είναι 91 % και το μικρότερο ποσοστό που έχει πετύχει είναι 80 %, τότε μπορούμε να υποθέσουμε ότι μπορούμε να φτάσουμε εύκολα σε αυτό το ποσοστό. Έτσι λοιπόν σε αυτές τις διαστάσεις είναι καλύτερα να χρησιμοποιούμε τις ΤΜΔ μόνο εάν χρησιμοποιούμε πρώτα επεξεργασία και μετά συμπύκνωση, γιατί αλλιώς ο θόρυβος μπορεί να επηρεάσει σε τεράστιο βαθμό τις ΤΜΔ. Όμως σε σύνολα με μικρά επίπεδα θορύβου ακόμα και μια ΤΜΔ μόνο με συμπύκνωση μπορεί να είναι πιο αποτελεσματική από τις μεθόδους δεικτοδότησης.

Όταν η διάσταση είναι μεταξύ 15 και 19 και σε εξάρτηση από την δομή του κάθε συνόλου μπορεί ή όχι το απλό brute force να είναι πιο αποτελεσματικό από τις μεθόδου δεικτοδότησης. Στα σύνολα που εξετάσαμε μέχρι τώρα δεν συμβαίνει κάτι τέτοιο βέβαια, πάρα μόνο κατά την επιλογή χαρακτηριστικών. Σε αυτές τις διαστάσεις είναι πιο αποτελεσματικό να χρησιμοποιήσουμε κάποια ΤΜΔ παρά τις μεθόδους δεικτοδότησης. Το μόνο θέμα που θα μας απασχολήσει είναι αν θα πρέπει να χρησιμοποιήσουμε πρώτα επεξεργασία ή όχι. Αν χρησιμοποιήσουμε ε-

πεξεργασία για να αφαιρέσουμε τον θόρυβο τότε σίγουρα θα έχουμε καλύτερα αποτελέσματα. Εάν όμως χρησιμοποιήσουμε μόνο συμπύκνωση υπάρχει το ενδεχόμενο οι μέθοδοι δεικτοδότησης να είναι και πιο γρήγορες και με μεγαλύτερη ακρίβεια, γιατί αν και έχουμε δυο σύνολα των ίδιων ακριβώς διαστάσεων (16) χρειαζόμαστε διαφορετικά ποσοστά μείωσης στο κάθε ένα. Επίσης όσο πλησιάζουμε τις 19 διαστάσεις χρειαζόμαστε να πετύχουμε μικρότερο ποσοστό μείωσης για να είναι καλύτερες οι ΤΜΔ.

Όταν η διάσταση είναι τουλάχιστον 20 τότε οι μέθοδοι δεικτοδότησης είναι πια αναποτελεσματικοί. Αυτό μας το επισημάνει το ίδιο το scikit-learn άλλα και επιβεβαιώνεται από τα πειράματα μας. Το kd-tree που μέχρι τώρα ήταν η καλύτερη μέθοδος δεικτοδότησης έχει χειρότερα αποτελέσματα από το ball-tree. Και οι δυο μέθοδοι δεικτοδότησης όμως είναι χειρότερες από το brute force. Σε αυτές τις διαστάσεις λοιπόν, καλύτερη μέθοδός είναι κάποια από τις ΤΜΔ.

Από τα δυο πειράματα βγάζουμε το συμπέρασμα ότι είναι καλύτερο να χρησιμοποιήσουμε κάποια ΤΜΔ, ειδικά αν χρησιμοποιούμε πρώτα επεξεργασία για αφαίρεση του θορύβου, όταν ένα σύνολο έχει περισσότερα από 10 χαρακτηριστικά. Αν και οι μέθοδοι δεικτοδότησης μπορεί να είναι γενικά αποτελεσματικοί μέχρι τις 20 διαστάσεις, δεν μπορούμε να είμαστε εντελώς σίγουροι πριν να κάνουμε κάποια δοκιμή. Έτσι είναι πιο ασφαλές να χρησιμοποιείται κάποια ΤΜΔ ειδικά όσο πλησιάζουμε στις 20 διαστάσεις. Για λιγότερες από 10 διαστάσεις είναι πιο ασφαλές να χρησιμοποιείται το kd-tree ως μέθοδος δεικτοδότησης με εξαίρεση αν είναι σίγουρο ότι μια ΤΜΔ μπορεί να πετύχει ποσοστό μείωσης μεταξύ 90-99 % ανάλογα με την διάσταση, όμως αυτό δεν είναι τόσο συχνό φαινόμενο.

Μιας και η παρούσα πτυχιακή εργασία ασχολείστε κυρίως με τους διανυσματικούς χώρους. Ως μελλοντική έρευνα θα ήταν ενδιαφέρουσα η μελέτη των γενικών μετρικών χωρών για αναζήτηση εγγυτέρων γειτόνων με ή χωρίς την χρήση μεθόδων δεικτοδότησης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] S. Ougiaroglou, *Algorithms and Techniques for Efficient and Effective Nearest Neighbours Classification*. PhD thesis, University of Macedonia, 2014.

[2] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Comput. Surv.*, vol. 33, pp. 273–321, Sept. 2001.

[3] C. Esperança, "Omohundro balltree construction algorithms," Feb 2021.

[4] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[5] "scikit-learn weights." `https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html`. Accessed: 2021-06-11.

[6] "scikit-learn model selection." `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection`. Accessed: 2021-06-11.

[7] M. James, *Classification algorithms*. New York, NY, USA: Wiley-Interscience, 1985.

[8] L. Rokach, *Data Mining with Decision Trees: Theory and Applications*. Series in machine perception and artificial intelligence, World Scientific Publishing Company, Incorporated, 2007.

[9] G. P. Zhang, "Neural networks for classification: A survey," *Trans. Sys. Man Cyber Part C*, vol. 30, pp. 451–462, Nov. 2000.

[10] H. Zhang, "The optimality of naive bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA* (V. Barr and Z. Markov, eds.), pp. 562–567, AAAI Press, 2004.

[11] R. Roiger and M. W. Geatz, *Data Mining: A Tutorial Based Primer*. Addison Wesley, 2003.

[12] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, pp. 21–27, Sept. 2006.

[13] B. V. Dasarathy, *Nearest neighbor (NN) norms : NN pattern classification techniques*. IEEE Computer Society Press, 1991.

[14] J. Kolodner, *Case-based Reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[15] S. Ougiaroglou, A. Nanopoulos, A. N. Papadopoulos, Y. Manolopoulos, and T. Welzer-Druzovec, "Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors," in *Proceedings of the 11th East European conference on Advances in databases and information systems*, ADBIS'07, (Berlin, Heidelberg), pp. 66–82, Springer-Verlag, 2007.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov. 2009.

[17] E. Deza and M. M. Deza, *Encyclopedia of Distances*. Berlin, Heidelberg: Springer, 2009.

[18] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-6, no. 4, pp. 325–327, 1976.

[19] J. L. Bentley, "A survey of techniques for fixed radius near neighbor searching.," tech. rep., 1975.

[20] H. Samet, *Foundations of multidimensional and metric data structures*. The Morgan Kaufmann series in computer graphics, Elsevier/Morgan Kaufmann, 2006.

[21] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search - The Metric Space Approach*, vol. 32. Springer, 2006.

[22] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, (San Francisco, CA, USA), pp. 194–205, Morgan Kaufmann Publishers Inc., 1998.

[23] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, pp. 417–435, Mar. 2012.

[24] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *Trans. Sys. Man Cyber Part C*, vol. 42, pp. 86–100, Jan. 2012.

[25] E. Chávez, G. Navarro, R. Baeza-Yates, and J. N, "Proximity searching in metric spaces," *ACM Computing Surveys - CSUR*, 01 2001.

[26] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, Dec. 2017.

[27] B. Chazelle, "Computational geometry: A retrospective," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, (New York, NY, USA), p. 75–94, Association for Computing Machinery, 1994.

[28] L. Amsaleg, O. Chelly, T. Furon, S. Girard, M. E. Houle, K.-i. Kawarabayashi, and M. Nett, "Estimating local intrinsic dimensionality," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, (New York, NY, USA), p. 29–38, Association for Computing Machinery, 2015.

[29] M. A. A. Cox and T. F. Cox, *Multidimensional Scaling*, pp. 315–347. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[30] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, Sept. 1975.

[31] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, pp. 209–226, Sept. 1977.

[32] *Orthogonal Range Searching*, pp. 95–120. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[33] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw Hill, 1990.

[34] "scikit-learn github." `https://github.com/scikit-learn/scikit-learn`. Accessed: 2021-06-13.

[35] A. Andoni, P. Indyk, and I. Razenshteyn, "Approximate nearest neighbor search in high dimensions," 2018.

[36] P. Indyk, "Nearest neighbors in high-dimensional spaces," 2004.

[37] S. M. Omohundro, "Five balltree construction algorithms," tech. rep., 1989.

[38] T. Liu, A. Moore, A. Gray, and P. Kaelbling, "New algorithms for efficient high-dimensional nonparametric," 05 2004.

[39] N. Kumar, L. Zhang, and S. Nayar, "What is a good nearest neighbors algorithm for finding similar patches in images," in *In ECCV*, 2008.

[40] J. S. Sánchez, "High training set size reduction by space partitioning and prototype abstraction," *Pattern Recognition*, vol. 37, no. 7, pp. 1561–1564, 2004.

[41] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE trans. on systems, man, and cybernetics*, vol. 2, pp. 408–421, July 1972.

[42] B. V. Dasarathy, J. S. Sánchez, and S. Townsend, "Nearest neighbour editing and condensing tools–synergy exploitation," *Pattern Analysis & Applications*, vol. 3, no. 1, pp. 19–30, 2000.

[43] J. Alcala-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. G. i Guiu, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernandez, and F. Herrera, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, pp. 307–318, Oct. 2008.

[44] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-basedlearning algorithms," *Mach. Learn.*, vol. 38, pp. 257–286, Mar. 2000.

[45] J. Sánchez, F. Pla, and F. Ferri, "Prototype selection for the nearest neighbour rule through proximity graphs," *Pattern Recognition Letters*, vol. 18, no. 6, pp. 507 – 513, 1997.

[46] F. Vázquez, J. S. Sánchez, and F. Pla, "A stochastic approach to wilson's editing algorithm," in *Proceedings of the Second Iberian conference on Pattern Recognition and Image Analysis - Volume Part II*, IbPRIA'05, (Berlin, Heidelberg), pp. 35–42, Springer-Verlag, 2005.

[47] K. Hattori and M. Takahashi, "A new edited k-nearest neighbor rule in the pattern classification problem," *Pattern Recognition*, vol. 33, no. 3, pp. 521 – 528, 2000.

[48] M. García-Borroto, Y. Villuendas-Rey, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "Using maximum similarity graphs to edit nearest neighbor classifiers," in *Proceedings of the 14th Iberoamerican Conference on Pattern Recognition: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, CIARP '09, (Berlin, Heidelberg), pp. 489–496, Springer-Verlag, 2009.

[49] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[50] C. Aggarwal, *Data Streams: Models and Algorithms*. Advances in Database Systems Series, Springer Science+Business Media, LLC, 2007.

[51] E. Mirkes, "Knn and potential energy: applet," 2011.

[52] Y. Shi, "Comparing k-nearest neighbors and potential energy method in classification problem. a case study using knn applet by e.m. mirkes and real life benchmark data sets," 2012.

[53] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, pp. 37–66, Jan. 1991.

[54] C. H. Chen and A. Jóźwik, "A sample set condensation algorithm for the class sensitive artificial neural network," *Pattern Recogn. Lett.*, vol. 17, pp. 819–823, July 1996.

[55] "Welcome to python.org." `https://www.python.org/about/`. Accessed: 2021-06-11.

[56] "scikit-learn." `https://scikit-learn.org/`. Accessed: 2021-06-11.

[57] “Python release history.” https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html. Accessed: 2021-06-11.

[58] “The making of python.” https://www.artima.com/articles/the-making-of-python. Accessed: 2021-06-11.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[60] “scikit-learn neighbors.” https://scikit-learn.org/stable/modules/neighbors.html#neighbors. Accessed: 2021-06-11.

[61] “Benchmarking nearest neighbor searches in python.” https://jakevdp.github.io/blog/2013/04/29/benchmarking-nearest-neighbor-searches-in-python/. Accessed: 2021-06-17.

[62] “scikit-learn preprocessing.” https://scikit-learn.org/stable/modules/preprocessing.html. Accessed: 2021-06-11.

[63] “scikit-learn metrics.” https://scikit-learn.org/stable/modules/model_evaluation.html. Accessed: 2021-06-11.

[64] “Uci machine learning repository.” https://archive.ics.uci.edu/ml/index.php. Accessed: 2021-06-17.

[65] M. Trabelsi, N. Meddouri, and M. Maddouri, “A new feature selection method for nominal classifier based on formal concept analysis,” *Procedia Computer Science*, vol. 112, pp. 186–194, 2017. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.