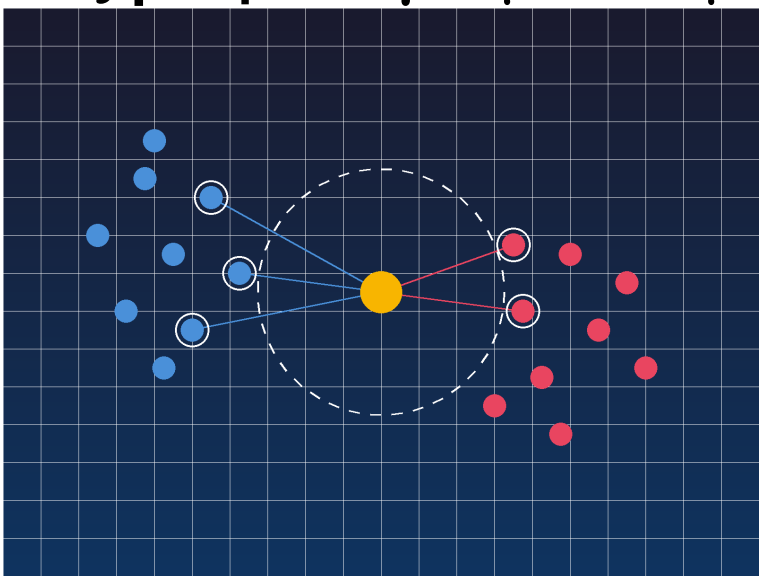
Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάλυση και Σύγκριση των Κατηγοριοποιητών kNN, Symmetric kNN και Mutual kNN και Ανάπτυξη Παραλλαγών με Δυναμικό k



Φοιτητής:
Ιωάννης Παπαδόπουλος
Αριθμός Μητρώου: 215273241585

Επιβλέπων:
Στέφανος Ουγιάρογλου

11 Φεβρουαρίου 2026

Title of Dissertation Ανάλυση και Σύγκριση των Κατηγοριοποιητών kNN, Symmetric kNN και Mutual kNN και Ανάπτυξη Παραλλαγών με Δυναμικό k

Code of Dissertation 25298

Student's full name Ioannis Papadopoulos

Supervisor's full name Stefanos Ougiaroglou

Date of undertaking 15-9-2025

Date of completion 11-02-2026

We hereby affirm the authorship of this paper as well as the acknowledgement and credit of whichever assistance We received in its composition. We have, furthermore, noted the various sources from which We extracted data, ideas, visual or written material, in paraphrase or exact quotation. Moreover, we affirm the exclusive composition of this paper by myself only, for the purpose of it being a dissertation, in the Department of Information and Electronic Engineering of the I.H.U.

The approval of this dissertation by the Department of Information and Electronic Engineering of the International Hellenic University, does not necessarily entail the adoption of the author's views, on behalf of the Department.

Αφιέρωση

Στους γονείς μου,
για την αμέριστη αγάπη, υποστήριξη και υπομονή τους
καθ' όλη τη διάρκεια των σπουδών μου.

Στους καθηγητές μου,
για την καθοδήγηση και τις πολύτιμες γνώσεις
που μου μετέδωσαν.

Abstract

This thesis addresses the in-depth study, implementation, and experimental evaluation of k-Nearest Neighbors (k-NN) algorithms and their modern variants for classification problems. The classic k-NN, despite its simplicity, remains one of the most popular machine learning algorithms due to its effectiveness across various application domains. However, it presents specific limitations, such as sensitivity to data noise, the use of a fixed number of neighbors k for all samples regardless of their local structure, and the absence of mechanisms for filtering unstable neighbor relationships. In this work, a comprehensive implementation was developed for the Conventional k-NN, Mutual k-NN—which is based on mutual neighbor relationships—and Symmetric k-NN—which applies symmetric union logic of neighbors—as well as the adaptive AdaNN algorithm that dynamically adjusts the number of neighbors per sample. Additionally, two new hybrid algorithms were proposed and implemented: Ada Mutual k-NN and Ada Symmetric k-NN, which combine the adaptability of AdaNN with the concepts of mutuality and symmetry respectively, aiming to improve classification accuracy through filtering unstable neighbor relationships and dynamic adjustment of k. An extensive experimental comparison of all methods was conducted on 14 well-known datasets from the UCI Machine Learning Repository and the KEEL Repository, examining their behavior under different conditions including varying data characteristics (number of classes, dimensions, dataset size) and the presence of 30% noise. The Friedman test confirmed the existence of statistically significant differences between methods both in noise-free data ($p = 4.17 \times 10^{-10}$) and in data with 30% noise ($p = 4.15 \times 10^{-38}$), indicating that the choice of KNN method significantly affects performance. On clean data, the Mutual KNN methods with adaptive k and lastk window=3 rank highest (mean rank 6.39), followed by the best k methods for Mutual and Conventional KNN (6.68 and 6.71 respectively), while on data with 30% noise, the static best k methods for Conventional and Symmetric KNN emerge as the best (4.50 and 4.61), suggesting that the presence of noise favors tuned selection of k per dataset over adaptive approaches. The Wilcoxon signed-rank test revealed that on clean data, statistically significant differences appear mainly in the Conventional vs Symmetric comparison for adaptive methods, while with the introduction of 30% noise, an increase in statistically significant differences is observed, especially for the Conventional vs Mutual comparison, suggesting that noise further differentiates method performance with Conventional KNN prevailing more frequently in noisy environments. In conclusion, the results show that there is no single method that excels in all scenarios, but the choice of the appropriate approach depends on data characteristics and the presence or absence of noise, with adaptive methods being favored on clean data and static methods with optimized k being more effective in the presence of noise.

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την εις βάθος μελέτη, υλοποίηση και πειραματική αξιολόγηση αλγορίθμων k-Nearest Neighbors (k-NN) και των σύγχρονων παραλλαγών τους για προβλήματα ταξινόμησης. Ο κλασικός k-NN, παρά την απλότητά του, παραμένει ένας από τους πιο δημοφιλείς αλγορίθμους μηχανικής μάθησης λόγω της αποτελεσματικότητάς του σε ποικίλα πεδία εφαρμογών. Ωστόσο, παρουσιάζει συγκεκριμένους περιορισμούς, όπως η ευαισθησία στον θόρυβο των δεδομένων, η χρήση σταθερού αριθμού γειτόνων k για όλα τα δείγματα ανεξαρτήτως της τοπικής τους δομής, και η απουσία μηχανισμών φιλτραρίσματος ασταθών γειτονικών σχέσεων. Στην παρούσα εργασία αναπτύχθηκε ολοκληρωμένη υλοποίηση των αλγορίθμων Conventional k-NN, Mutual k-NN—ο οποίος βασίζεται σε αμοιβαίες γειτονικές σχέσεις—και Symmetric k-NN—ο οποίος εφαρμόζει συμμετρική λογική ένωσης γειτόνων—καθώς και του adaptive αλγορίθμου AdaNN που προσαρμόζει δυναμικά τον αριθμό γειτόνων ανά δείγμα. Επιπλέον, προτάθηκαν και υλοποιήθηκαν δύο νέοι υβριδικοί αλγόριθμοι, ο Ada Mutual k-NN και ο Ada Symmetric k-NN, οι οποίοι συνδυάζουν την προσαρμοστικότητα του AdaNN με τις έννοιες της αμοιβαιότητας και της συμμετρίας αντίστοιχα, στοχεύοντας στη βελτίωση της ακρίβειας ταξινόμησης μέσω του φιλτραρίσματος ασταθών γειτονικών σχέσεων και της δυναμικής προσαρμογής του k. Πραγματοποιήθηκε εκτενής πειραματική σύγκριση όλων των μεθόδων σε 14 γνωστά datasets από το UCI Machine Learning Repository και το KEEL Repository, εξετάζοντας τη συμπεριφορά τους υπό διαφορετικές συνθήκες που περιλαμβάνουν μεταβαλλόμενα χαρακτηριστικά δεδομένων (πλήθος κλάσεων, διαστάσεις, μέγεθος συνόλου) και παρουσία θορύβου 30%. Το Friedman test επιβεβαίωσε την ύπαρξη στατιστικά σημαντικών διαφορών μεταξύ των μεθόδων τόσο στα δεδομένα χωρίς θόρυβο ($p = 4.17\times10^{-10}$) όσο και στα δεδομένα με θόρυβο 30% ($p = 4.15\times10^{-38}$), υποδεικνύοντας ότι η επιλογή της μεθόδου KNN επηρεάζει σημαντικά την απόδοση. Στα καθαρά δεδομένα, οι μέθοδοι Mutual KNN με adaptive k και lastk window=3 κατατάσσονται υψηλότερα (μέση κατάταξη 6.39), ακολουθούμενες από τις μεθόδους best k για Mutual και Conventional KNN (6.68 και 6.71 αντίστοιχα), ενώ στα δεδομένα με θόρυβο 30%, οι στατικές μέθοδοι best k για Conventional και Symmetric KNN αναδεικνύονται ως οι καλύτερες (4.50 και 4.61), υποδηλώνοντας ότι η παρουσία θορύβου ευνοεί την προσαρμοσμένη επιλογή του k ανά dataset έναντι των adaptive προσεγγίσεων. Το Wilcoxon signed-rank test αποκάλυψε ότι στα καθαρά δεδομένα οι στατιστικά σημαντικές διαφορές εμφανίζονται κυρίως στη σύγκριση Conventional vs Symmetric για τις adaptive μεθόδους, ενώ με την εισαγωγή θορύβου 30% παρατηρείται αύξηση των στατιστικά σημαντικών διαφορών, ιδίως για τη σύγκριση Conventional vs Mutual, υποδηλώνοντας ότι ο θόρυβος διαφοροποιεί περισσότερο την απόδοση των μεθόδων με την Conventional KNN να υπερτερεί συχνότερα σε θορυβώδη περιβάλλοντα. Συμπερασματικά, τα αποτελέσματα δείχνουν ότι δεν υπάρχει μία μέθοδος που υπερτερεί σε όλα τα σενάρια, αλλά η επιλογή της κατάλληλης προσέγγισης εξαρ-

τάται από τα χαρακτηριστικά των δεδομένων και την παρουσία ή απουσία θορύβου, με τις adaptive μεθόδους να ευνοούνται σε καθαρά δεδομένα και τις στατικές μεθόδους με βελτιστοποιημένο k να είναι πιο αποτελεσματικές παρουσία θορύβου.

Περιεχόμενα

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Eager vs Lazy/Instance-Based Learning

Οι αλγόριθμοι μηχανικής μάθησης κατηγοριοποιούνται με διάφορους τρόπους ανάλογα με τα χαρακτηριστικά τους. Μία σημαντική διάκριση αφορά τον τρόπο με τον οποίο μαθαίνουν από τα δεδομένα και πότε πραγματοποιούν τις προβλέψεις τους.
Οι αλγόριθμοι μηχανικής μάθησης διακρίνονται σε δύο κύριες κατηγορίες ανάλογα με τον τρόπο που χειρίζονται τη διαδικασία της εκπαίδευσης και της πρόβλεψης, με σημαντικές διαφορές στο πότε και πώς επενδύεται το υπολογιστικό κόστος.
Οι **Eager Learners** (πρόθυμοι μαθητές) κατασκευάζουν ένα γενικευμένο μοντέλο κατά τη φάση της εκπαίδευσης, πριν δουν οποιοδήποτε δείγμα δοκιμής. Αυτοί οι αλγόριθμοι επενδύουν σημαντικό υπολογιστικό κόστος και χρόνο κατά την εκπαίδευση προκειμένου να χτίσουν μια εσωτερική αναπαράσταση ή μοντέλο των δεδομένων που συλλαμβάνει τις υποκείμενες σχέσεις και μοτίβα. Παραδείγματα τέτοιων αλγορίθμων περιλαμβάνουν τα δέντρα απόφασης, τα νευρωνικά δίκτυα και τα Support Vector Machines. Το κύριο πλεονέκτημα αυτής της προσέγγισης είναι ότι, αφού κατασκευαστεί το μοντέλο, η φάση της πρόβλεψης είναι εξαιρετικά γρήγορη και αποδοτική, καθώς απαιτεί απλώς την εφαρμογή του προ-υπολογισμένου μοντέλου στα νέα δεδομένα. Επιπλέον, το μοντέλο που προκύπτει μπορεί να προσφέρει ερμηνευσιμότητα και γενίκευση, επιτρέποντας την κατανόηση των συνολικών τάσεων και δομών στα δεδομένα.
Οι **Lazy Learners** ή **Instance-Based Learners** (οκνηροί μαθητές ή μαθητές βασισμένοι σε περιπτώσεις) ακολουθούν μια θεμελιωδώς διαφορετική φιλοσοφία. Αντί να κατασκευάζουν ένα γενικευμένο μοντέλο κατά την εκπαίδευση, αποθηκεύουν απλώς τα εκπαιδευτικά δεδομένα στη μνήμη και αναβάλλουν όλη την επεξεργασία μέχρι τη στιγμή που λάβουν ένα νέο δείγμα προς ταξινόμηση ή πρόβλεψη. Η βασική ιδέα είναι ότι βασίζονται άμεσα στα συγκεκριμένα αποθηκευμένα δείγματα (instances) για να λάβουν αποφάσεις, χωρίς να χρειάζεται να εξάγουν κανόνες ή να μαθαίνουν παραμέτρους που περιγράφουν τη συνολική δομή των δεδομένων.
Τα χαρακτηριστικά των lazy/instance-based learners είναι ιδιαίτερα διακριτά και περιλαμβάνουν πολλές σημαντικές ιδιότητες. Πρώτον, η φάση εκπαίδευσης είναι εξαιρετικά γρήγορη και απλή, καθώς περιορίζεται κυρίως στην αποθήκευση των δεδομένων και πιθανώς στην κατασκευή κάποιων βοηθητικών δομών δεδομένων για την επιτάχυνση της αναζήτησης, όπως k-d trees ή ball trees. Δεύτερον, οι απαιτήσεις μνήμης είναι υψηλές και ανάλογες του μεγέθους του εκπαιδευτικού συνόλου, καθώς πρέπει να διατηρηθούν όλα ή ένα μεγάλο υποσύνολο των εκπαιδευτικών δειγμάτων. Τρίτον, η πρόβλεψη βασίζεται στην έννοια της τοπικής ομοιότητας,

πράγμα που σημαίνει ότι για κάθε νέο δείγμα εξετάζονται μόνο τα πλησιέστερα αποθηκευμένα δείγματα στο χώρο χαρακτηριστικών, επιτρέποντας στον αλγόριθμο να προσαρμόζεται σε τοπικές διακυμάνσεις και ιδιαιτερότητες της κατανομής των δεδομένων. Τέταρτον, οι αλγόριθμοι αυτοί προσαρμόζονται εύκολα σε νέα δεδομένα μέσω incremental learning, καθώς η προσθήκη νέων εκπαιδευτικών δειγμάτων απλώς επεκτείνει το αποθηκευμένο σύνολο χωρίς να απαιτείται η δαπανηρή επανεκπαίδευση ολόκληρου μοντέλου. Πέμπτον, το υπολογιστικό κόστος μετατοπίζεται από την εκπαίδευση στην πρόβλεψη, καθώς κάθε νέο δείγμα απαιτεί τον υπολογισμό αποστάσεων ή ομοιοτήτων προς πολλά ή όλα τα αποθηκευμένα εκπαιδευτικά δείγματα, κάτι που μπορεί να είναι υπολογιστικά δαπανηρό ειδικά για μεγάλα σύνολα δεδομένων με υψηλή διαστατικότητα.
Ο αλγόριθμος k-NN (k-Nearest Neighbors) και οι πολυάριθμες παραλλαγές του, όπως ο AdaNN, ο Mutual k-NN, ο Symmetric k-NN και άλλες προσαρμοστικές μέθοδοι, αποτελούν τα πιο χαρακτηριστικά και ευρέως χρησιμοποιούμενα παραδείγματα lazy/instance-based learners. Αυτοί οι αλγόριθμοι δεν χτίζουν κανένα μοντέλο κατά την εκπαίδευση και βασίζονται αποκλειστικά στην άμεση σύγκριση με τα αποθηκευμένα εκπαιδευτικά δείγματα κατά την πρόβλεψη. Κάθε απόφαση ταξινόμησης λαμβάνεται τοπικά, εξετάζοντας μόνο την άμεση γειτονιά του δείγματος προς ταξινόμηση, και η ποιότητα της πρόβλεψης εξαρτάται άμεσα από την πυκνότητα και την αντιπροσωπευτικότητα των εκπαιδευτικών δειγμάτων στην εκάστοτε περιοχή του χώρου χαρακτηριστικών. Αυτή η τοπική φύση των αποφάσεων καθιστά τους instance-based learners ιδιαίτερα κατάλληλους και αποτελεσματικούς για προβλήματα με πολύπλοκες, μη γραμμικές δομές και ανομοιόμορφες κατανομές κλάσεων, όπου η εξαγωγή ενός καθολικού παραμετρικού μοντέλου θα ήταν δύσκολη, αναποτελεσματική ή θα απαιτούσε υπερβολική πολυπλοκότητα.

1.2 Κατηγοριοποίηση k εγγύτερων γειτόνων

Ο αλγόριθμος k-Nearest Neighbors (k-NN) [1] αποτελεί μία από τις πιο θεμελιώδεις και διαισθητικές μεθόδους στη μηχανική μάθηση, ανήκοντας στην κατηγορία της "τεμπέλικης μάθησης" (lazy learning), καθώς δεν κατασκευάζει ένα ρητό μοντέλο κατά τη φάση της εκπαίδευσης, αλλά απομνημονεύει τα δεδομένα και διενεργεί υπολογισμούς μόνο κατά τη στιγμή της πρόβλεψης. Η κεντρική ιδέα βασίζεται στην υπόθεση ότι παρόμοια δεδομένα βρίσκονται κοντά το ένα στο άλλο μέσα στον διανυσματικό χώρο των χαρακτηριστικών.
Για να ταξινομήσει ένα νέο, άγνωστο δείγμα, ο αλγόριθμος υπολογίζει την απόσταση του δείγματος αυτού από όλα τα αποθηκευμένα σημεία του συνόλου εκπαίδευσης, χρησιμοποιώντας συνήθως την Ευκλείδεια απόσταση (Euclidean distance) ή άλλες μετρικές όπως η απόσταση Manhattan ή Minkowski, ανάλογα με τη φύση των δεδομένων. Στη συνέχεια, επιλέγει τα k πλησιέστερα σημεία (τους "γείτονες") και αναθέτει στο νέο δείγμα την κλάση που κυριαρχεί μεταξύ αυτών των γειτόνων, μέσω μιας διαδικασίας πλειοψηφίας (majority voting).
Η επιλογή της παραμέτρου k είναι κρίσιμη: ένα πολύ μικρό k (π.χ. $k = 1$) καθιστά τον αλγόριθμο εξαιρετικά ευαίσθητο στο θόρυβο και τα μεμονωμένα ακραία σημεία (outliers), οδηγώντας σε overfitting, ενώ ένα πολύ μεγάλο k μπορεί να εξομαλύνει υπερβολικά τα όρια απόφασης, συμπεριλαμβάνοντας γείτονες από μακρινές και άσχετες κλάσεις, οδηγώντας σε underfitting. Παρά την απλότητά του, ο k-NN αντιμετωπίζει προκλήσεις όπως η "κατάρα της διαστατικότητας" (curse of dimensionality), όπου σε χώρους πολλών διαστάσεων η έννοια της εγγύτητας χάνει το νόημά της καθώς όλα τα σημεία τείνουν να απέχουν εξίσου μεταξύ τους.

Πλεονεκτήματα:
Η απλότητα και διαισθητικότητα του αλγορίθμου τον καθιστούν εύκολο στην κατανόηση και υλοποίηση. Δεν απαιτεί καμία υπόθεση σχετικά με την κατανομή των δεδομένων (μη παραμετρική μέθοδος), και μπορεί να προσαρμοστεί σε πολύπλοκα όρια απόφασης. Επιπλέον, δεν υπάρχει φάση εκπαίδευσης (lazy learning), καθώς όλοι οι υπολογισμοί γίνονται κατά την πρόβλεψη, και ο αλγόριθμος είναι φυσικά πολυκλασσικός, δηλαδή μπορεί να χειριστεί προβλήματα με περισσότερες από δύο κλάσεις χωρίς τροποποιήσεις. Τέλος, προσαρμόζεται άμεσα σε νέα δεδομένα με απλή προσθήκη τους στο σύνολο εκπαίδευσης.
Περιορισμοί:
Η υπολογιστική πολυπλοκότητα κατά την πρόβλεψη είναι $O(nd)$, όπου απαιτείται ο υπολογισμός αποστάσεων από όλα τα n σημεία εκπαίδευσης σε d διαστάσεις, κάτι που τον καθιστά αργό σε μεγάλα σύνολα δεδομένων. Επιπλέον, απαιτεί μεγάλη μνήμη για την αποθήκευση ολόκληρου του συνόλου εκπαίδευσης. Ο αλγόριθμος είναι ευαίσθητος στην κλίμακα των χαρακτηριστικών, καθιστώντας απαραίτητη την κανονικοποίηση, και υποφέρει από την «κατάρα της διαστατικότητας» (curse of dimensionality): σε υψηλοδιάστατους χώρους, η έννοια της «εγγύτητας» χάνει το νόημά της, καθώς όλα τα σημεία τείνουν να βρίσκονται σε παρόμοιες αποστάσεις μεταξύ τους. Επίσης, είναι ευάλωτος στο θόρυβο και τις ανισορροπημένες κατανομές κλάσεων, ενώ δεν μπορεί να προσφέρει επεξηγησιμότητα σχετικά με τη σημαντικότητα των χαρακτηριστικών.
Για την αντιμετώπιση αυτών των περιορισμών, έχουν αναπτυχθεί διάφορες βελτιωμένες παραλλαγές, όπως οι δομές δεδομένων KD-trees και Ball-trees για ταχύτερη αναζήτηση γειτόνων, οι τεχνικές επιλογής χαρακτηριστικών και μείωσης διαστατικότητας, καθώς και η εφαρμογή της μεθόδου SMOTE για την αντιμετώπιση ανισορροπημένων συνόλων δεδομένων.

1.3 Παραμετροποίηση του k

Η επιλογή της κατάλληλης τιμής για την παράμετρο k αποτελεί κρίσιμο βήμα για την επιτυχή εφαρμογή του αλγορίθμου k-NN. Υπάρχουν διάφορες τεχνικές που μπορούν να χρησιμοποιηθούν για αυτόν το σκοπό, η καθεμία με τα δικά της πλεονεκτήματα και περιορισμούς. Ο κανόνας της τετραγωνικής ρίζας ($k = \sqrt{n}$) παρέχει μια απλή και γρήγορη εκτίμηση, ενώ η διασταυρούμενη επικύρωση (k-fold cross-validation) θεωρείται η πιο αξιόπιστη μέθοδος καθώς αξιολογεί συστηματικά την απόδοση του μοντέλου σε διαφορετικά υποσύνολα των δεδομένων. Η αναζήτηση πλέγματος (Grid Search) προσφέρει μια εξαντλητική προσέγγιση δοκιμάζοντας συστηματικά ένα εύρος υποψήφιων τιμών, συχνά σε συνδυασμό με διασταυρούμενη επικύρωση. Παράλληλα, είναι απαραίτητο να λαμβάνονται υπόψη συγκεκριμένα κριτήρια επιλογής που αφορούν στην ισορροπία μεταξύ υπερπροσαρμογής και υποπροσαρμογής, καθώς και πρακτικές συστάσεις που διευκολύνουν τη διαδικασία βελτιστοποίησης της παραμέτρου.
Κανόνας της Τετραγωνικής Ρίζας. Μια απλή ευρετική μέθοδος για την επιλογή της παραμέτρου k είναι η χρήση του τύπου $k = \sqrt{n}$, όπου n είναι το πλήθος των δειγμάτων εκπαίδευσης. Συνήθως επιλέγουμε περιττό k για να αποφύγουμε ισοπαλίες στην ταξινόμηση.
Διασταυρούμενη Επικύρωση. Η πιο αξιόπιστη τεχνική για την επιλογή του k είναι η k-fold cross-validation. Σύμφωνα με αυτή τη μέθοδο, διαιρούμε το σύνολο δεδομένων σε m υποσύνολα. Για κάθε υποψήφια τιμή του k (π.χ. $k \in \{1, 3, 5, 7, \ldots, 21\}$), εκπαιδεύουμε το μοντέλο σε $m-1$ υποσύνολα, αξιολογούμε την απόδοσή του στο υπόλοιπο υποσύνολο και υπολογίζουμε τη μέση

ακρίβεια. Τελικά, επιλέγουμε την τιμή του k που παρουσιάζει την καλύτερη μέση απόδοση σε όλα τα υποσύνολα επικύρωσης.

Αναζήτηση Πλέγματος. Η μέθοδος Grid Search δοκιμάζει συστηματικά ένα εύρος τιμών για την παράμετρο k, συνήθως περιττούς αριθμούς όπως $k \in \{1, 3, 5, 7, 9, \ldots, k_{\text{max}}\}$. Αυτή η προσέγγιση συνδυάζεται συνήθως με διασταυρούμενη επικύρωση για την αντικειμενική αξιολόγηση κάθε τιμής και την επιλογή της βέλτιστης παραμέτρου.

Κριτήρια Επιλογής. Κατά την επιλογή του k πρέπει να λαμβάνουμε υπόψη διάφορους παράγοντες που επηρεάζουν την απόδοση του αλγορίθμου. Οι μικρές τιμές του k (όπως $k = 1$ ή $k = 3$) είναι ιδιαίτερα ευαίσθητες στο θόρυβο των δεδομένων και μπορούν να οδηγήσουν σε υπερπροσαρμογή, δημιουργώντας πολύπλοκα όρια απόφασης που δεν γενικεύουν καλά σε νέα δεδομένα. Αντίθετα, οι μεγάλες τιμές του k (όπως $k > 20$) τείνουν να προκαλούν υποπροσαρμογή, απλοποιώντας υπερβολικά τα όρια απόφασης και χάνοντας σημαντικές λεπτομέρειες των δεδομένων. Είναι σημαντικό να επιλέγουμε περιττές τιμές για το k ώστε να αποφεύγουμε ισοπαλίες στη δυαδική ταξινόμηση. Επιπλέον, το υπολογιστικό κόστος αυξάνεται με μεγαλύτερες τιμές του k, καθώς απαιτούνται περισσότεροι υπολογισμοί αποστάσεων και συγκρίσεις κατά την πρόβλεψη.

1.4 Κίνητρα της εργασίας

Τα κύρια κίνητρα της παρούσας εργασίας προέκυψαν από την ανάγκη εις βάθος κατανόησης των αλγορίθμων k-NN και των σύγχρονων παραλλαγών τους, καθώς και από την επιθυμία συστηματικής αξιολόγησης της απόδοσής τους σε πραγματικά προβλήματα ταξινόμησης. Ο αλγόριθμος k-NN, παρά την απλότητά του, παραμένει ιδιαίτερα δημοφιλής λόγω της αποτελεσματικότητάς του σε ποικίλα πεδία εφαρμογών. Ωστόσο, υπάρχει συνεχής ενδιαφέρον για τη βελτίωση της απόδοσής του μέσω εξελιγμένων τεχνικών που αντιμετωπίζουν τους περιορισμούς του. Συγκεκριμένα, η ευαισθησία του κλασικού k-NN στον θόρυβο των δεδομένων, η σταθερότητα του αριθμού γειτόνων k για όλα τα δείγματα ανεξαρτήτως της τοπικής δομής τους, και η απουσία φιλτραρίσματος ασταθών γειτονικών σχέσεων, αποτελούν ζητήματα που έχουν προσελκύσει ερευνητικό ενδιαφέρον. Παράλληλα, παραλλαγές όπως ο Mutual k-NN και ο Symmetric k-NN προτείνουν διαφορετικές προσεγγίσεις στον ορισμό των γειτονικών σχέσεων, ενώ οι adaptive μέθοδοι επιτρέπουν την προσαρμογή του k ανά δείγμα. Η ένωση αυτών των τεχνικών παραμένει ένα ανοιχτό ερευνητικό πεδίο με πιθανές βελτιώσεις στην ακρίβεια και την ευρωστία. Επιπλέον, η συστηματική σύγκριση αυτών των μεθόδων σε διαφορετικά datasets και υπό διαφορετικές συνθήκες—όπως η παρουσία θορύβου—παραμένει περιορισμένη στη βιβλιογραφία. Τέλος, η διαθεσιμότητα εξειδικευμένων repositories όπως τα UCI Machine Learning Repository και KEEL, που παρέχουν ποιοτικά datasets για πειραματική αξιολόγηση, καθιστά δυνατή την αξιόπιστη σύγκριση και επικύρωση νέων μεθόδων.

1.5 Συνεισφορά της εργασίας

Η κύρια συνεισφορά της παρούσας εργασίας εντοπίζεται σε τρεις βασικούς άξονες. Πρώτον, αναπτύχθηκε μια ολοκληρωμένη υλοποίηση των αλγορίθμων k-NN, Mutual k-NN, Symmetric k-NN και AdaNN από την αρχή, με έμφαση στην αποδοτικότητα και την ακρίβεια του κώδικα,

επιτρέποντας την πλήρη κατανόηση των μηχανισμών λειτουργίας τους και την ευελιξία παραμετροποίησης. Δεύτερον, πραγματοποιήθηκε εκτενής πειραματική σύγκριση όλων των αλγορίθμων σε 14 γνωστά datasets από το UCI Machine Learning Repository και το KEEL Repository, εξετάζοντας τη συμπεριφορά τους υπό διαφορετικές συνθήκες όπως μεταβαλλόμενα χαρακτηριστικά δεδομένων (πλήθος κλάσεων, διαστάσεις, μέγεθος συνόλου) και παρουσία θορύβου σε ποικίλα ποσοστά. Η συστηματική αυτή αξιολόγηση παρέχει πολύτιμες πληροφορίες σχετικά με τα πλεονεκτήματα και τους περιορισμούς κάθε μεθόδου σε διαφορετικά πλαίσια εφαρμογής. Τρίτον και σημαντικότερον, αναπτύχθηκαν και υλοποιήθηκαν δύο νέοι υβριδικοί αλγόριθμοι—ο **Ada Mutual k-NN** και ο **Ada Symmetric k-NN**—οι οποίοι συνδυάζουν την προσαρμοστικότητα του AdaNN με τις έννοιες της αμοιβαιότητας και της συμμετρίας αντίστοιχα. Οι αλγόριθμοι αυτοί αποτελούν πρωτότυπες προτάσεις που στοχεύουν στη βελτίωση της ακρίβειας ταξινόμησης μέσω του φιλτραρίσματος ασταθών γειτονικών σχέσεων και της δυναμικής προσαρμογής του αριθμού γειτόνων ανά δείγμα. Η συνεισφορά αυτή επεκτείνει τη βιβλιογραφία των αλγορίθμων k-NN και παρέχει νέες προοπτικές για την αντιμετώπιση προβλημάτων κατηγοριοποίησης σε πολύπλοκα και θορυβώδη περιβάλλοντα.

1.6 Οργάνωση της Διατριβής

Η παρούσα διπλωματική εργασία οργανώνεται σε επτά κεφάλαια, τα οποία δομούνται ως εξής:

Στο **Κεφάλαιο 1** παρουσιάζεται μια εισαγωγή στους αλγορίθμους k-Nearest Neighbors, εξηγώντας τη διάκριση μεταξύ eager και lazy learning, τη βασική αρχή λειτουργίας του k-NN, και τις μεθόδους παραμετροποίησης του k. Επιπλέον, διατυπώνονται τα κίνητρα και η συνεισφορά της εργασίας.

Το **Κεφάλαιο 2** περιλαμβάνει μια εκτενή βιβλιογραφική ανασκόπηση των μεθόδων δυναμικού k για κατηγοριοποίηση, εστιάζοντας στον θεμελιώδη αλγόριθμο AdaNN, τις επεκτάσεις του, εναλλακτικές προσεγγίσεις, θεωρητικά θεμέλια, συγκριτικές αξιολογήσεις, και εφαρμογές σε ειδικούς τομείς.

Στο **Κεφάλαιο 3** περιγράφεται η προϋπάρχουσα γνώση που αποτελεί τη βάση της εργασίας, συμπεριλαμβανομένων των αλγορίθμων AdaNN, Mutual k-NN, και Symmetric k-NN, με λεπτομερή παρουσίαση των μηχανισμών λειτουργίας τους.

Το **Κεφάλαιο 4** παρουσιάζει τις πρωτότυπες προτάσεις της εργασίας: τους υβριδικούς αλγορίθμους Ada Mutual k-NN και Ada Symmetric k-NN, οι οποίοι συνδυάζουν την προσαρμοστικότητα του AdaNN με τις έννοιες της αμοιβαιότητας και της συμμετρίας, καθώς και τον αλγόριθμο Ada Last Stable k-NN που εισάγει μηχανισμό σταθεροποίησης του k.

Στο **Κεφάλαιο 5** περιγράφεται αναλυτικά η υλοποίηση των αλγορίθμων σε Python, συμπεριλαμβανομένων των χρησιμοποιούμενων βιβλιοθηκών, της οργάνωσης του κώδικα, και των βασικών συναρτήσεων για τον υπολογισμό του adaptive k, τη διασταυρούμενη επικύρωση, και τη στατιστική ανάλυση με τα Friedman και Wilcoxon tests.

Το **Κεφάλαιο 6** περιλαμβάνει την εκτενή πειραματική μελέτη, παρουσιάζοντας τα 14 datasets που χρησιμοποιήθηκαν από το UCI Machine Learning Repository και το KEEL Repository, τη μεθοδολογία των πειραμάτων, τα αποτελέσματα των μετρήσεων τόσο σε καθαρά δεδομένα όσο και σε δεδομένα με θόρυβο 30%, και τη στατιστική ανάλυση με τα Friedman και Wilcoxon signed-rank tests.

Τέλος, στο **Κεφάλαιο 7** συνοψίζονται τα συμπεράσματα της εργασίας, αναλύονται τα ευρήματα από τα στατιστικά πειράματα, και προτείνονται κατευθύνσεις για μελλοντική έρευνα στο πεδίο των αλγορίθμων k-NN και των adaptive μεθόδων.

Κεφάλαιο 2

Literature Review

2.1 Μέθοδοι δυναμικού k για κατηγοριοποίηση

Ο αλγόριθμος Adaptive k-Nearest Neighbor (AdaNN) αντιπροσωπεύει μια σημαντική εξέλιξη στη μάθηση βασισμένη σε περιπτώσεις, αντιμετωπίζοντας τον θεμελιώδη περιορισμό της επιλογής σταθερού μεγέθους γειτονιάς για όλα τα ερωτήματα. Όπως επισημαίνουν οι Papanikolaou, Evangelidis και Ougiaroglou [2] στην εκτενή βιβλιογραφική τους ανασκόπηση, η έρευνα για τη δυναμική επιλογή του k έχει οδηγήσει σε πολυάριθμες προσεγγίσεις που εκμεταλλεύονται γενετικούς αλγορίθμους, νευρωνικά δίκτυα, prototypes και clustering, ευρετικές μεθόδους, πιθανοτικές προσεγγίσεις και άλλες τεχνικές. Το θεμελιώδες άρθρο των Sun και Huang το 2010 εισήγαγε την επιλογή του k ανά δείγμα, γεννώντας πολυάριθμες επεκτάσεις σε θεωρητική ανάλυση, υπολογιστική βελτιστοποίηση και εφαρμογές σε διάφορους τομείς.

2.1.1 Ο θεμελιώδης αλγόριθμος AdaNN

Ο αρχικός αλγόριθμος AdaNN προέκυψε από την αναγνώριση ότι οι ομοιόμορφες τιμές k αποτυγχάνουν να καταγράψουν τις διακυμάνσεις της τοπικής δομής των δεδομένων. Οι Sun και Huang [3] πρότειναν τον προσδιορισμό ενός βέλτιστου k ξεχωριστά για κάθε δείγμα εκπαίδευσης ορισμένο ως τον ελάχιστο αριθμό γειτόνων που απαιτούνται για σωστή ταξινόμηση και στη συνέχεια την κληρονόμηση αυτής της τιμής για δείγματα δοκιμής με βάση τον πλησιέστερο εκπαιδευτικό γείτονά τους. Το άρθρο απέδειξε ότι το ποσοστό σφάλματος του AdaNN βρίσκεται μεταξύ του σφάλματος Bayes και του διπλάσιου σφάλματος Bayes, με την απόδοση στην καλύτερη περίπτωση να πλησιάζει το βέλτιστο σφάλμα Bayes. Όπως αναφέρουν οι Papanikolaou et al. [2], αυτή η εργασία με περίπου 103 αναφορές (σύμφωνα με τη μελέτη τους το 2021) καθιέρωσε τη θεμελιώδη ιδέα για την προσαρμοστική επιλογή γειτονιάς ανά δείγμα. Τα πειραματικά αποτελέσματα σε 15 datasets από το UCI Machine Learning Repository έδειξαν ότι ο AdaNN υπερτερεί των παραδοσιακών k-NN αλγορίθμων σε 11 από τα 15 σύνολα δεδομένων.

2.1.2 Άμεσες επεκτάσεις του πλαισίου AdaNN

Αρκετά άρθρα επεκτείνουν άμεσα τη μεθοδολογία των Sun και Huang με ουσιαστικές καινοτομίες. Οι Pan, Wang και Pan [4] εισήγαγαν τον Discrimination Class Locally Adaptive k-NN (DC-LAKNN), ο οποίος λαμβάνει υπόψη τόσο την πλειοψηφική κλάση όσο και τη δεύτερη σε

πλειοψηφία κλάση κατά τον προσδιορισμό των προσαρμοστικών τιμών k. Αυτή η προσέγγιση χρησιμοποιεί πληροφορίες ποσότητας και κατανομής από τις κλάσεις διάκρισης, επιτυγχάνοντας ανώτερη απόδοση σε 18 benchmark datasets από το UCI και το KEEL σε σύγκριση με εννέα άλλες παραλλαγές k-NN. Η καινοτομία έγκειται στην αναγνώριση ότι η σχέση μεταξύ των δύο κορυφαίων ανταγωνιστικών κλάσεων είναι κρίσιμη για τη βέλτιστη επιλογή του k, ειδικά κοντά στα όρια απόφασης.

Οι Anava και Levy [5] προώθησαν την προσαρμοστική επιλογή του k μέσω της ρητής διατύπωσης bias-variance στη μέθοδό τους k*-Nearest Neighbors. Σε αντίθεση με τη διακριτή επιλογή k του AdaNN, το k* παρέχει συνεχή βελτιστοποίηση με θεωρητικές εγγυήσεις, προσδιορίζοντας από κοινού τα βέλτιστα βάρη και το μέγεθος της γειτονιάς μέσω αρχών ανάλυσης. Η μέθοδος επιλύει ένα πρόβλημα βελτιστοποίησης που εξισορροπεί το σφάλμα προσέγγισης (bias) με τη διακύμανση εκτίμησης (variance), παρέχοντας ένα πιο θεωρητικά θεμελιωμένο πλαίσιο για την προσαρμοστική επιλογή γειτόνων. Σύμφωνα με τη μελέτη των Papanikolaou et al. [2], αυτή η εργασία έχει λάβει 50 αναφορές με μέσο όρο 10 αναφορές ανά έτος, υποδεικνύοντας τη σημαντική της επιρροή στο πεδίο.

Οι Di Salvo et al. [6] επέκτειναν τους προσαρμοστικούς πλησιέστερους γείτονες σε σενάρια βαθιάς μάθησης με θορυβώδεις ετικέτες μέσω της μεθόδου WANN (Weighted Adaptive Nearest Neighbor). Λειτουργώντας σε embeddings από μοντέλα βάσης (foundation models), το WANN αναδιατυπώνει την έννοια του βέλτιστου k του AdaNN ως μετρική αξιοπιστίας ετικετών, επιδεικνύοντας ανώτερη απόδοση υπό συμμετρικό, ασύμμετρο και εξαρτώμενο από το δείγμα θόρυβο. Αυτή η πρόσφατη εργασία (2024) δείχνει πώς οι αρχές του AdaNN μπορούν να προσαρμοστούν στα σύγχρονα προβλήματα μάθησης με θόρυβο.

Οι Mullick, Datta και Das [7] ανέπτυξαν δύο εκδοχές του AdaNN που χρησιμοποιούν τεχνητά νευρωνικά δίκτυα (Ada-k-NN) και ευρετικές μεθόδους (Ada-k-NN2) για τη μάθηση κατάλληλων τιμών k βασισμένες στην τοπική πυκνότητα και κατανομή των δεδομένων. Όπως αναφέρουν οι Papanikolaou et al. [2], το Ada-k-NN2 επέτυχε την καλύτερη μέση κατάταξη σε μικρά και μεσαίου μεγέθους datasets, ενώ και οι δύο μέθοδοι επέδειξαν ανταγωνιστική απόδοση σε προβλήματα με ανισορροπημένες κλάσεις όταν συνδυάστηκαν με κατάλληλα σχήματα στάθμισης.

2.1.3 Εναλλακτικές προσεγγίσεις για προσαρμοστική επιλογή k

Πέρα από τις άμεσες επεκτάσεις του AdaNN, αρκετά επιδραστικά άρθρα προτείνουν διακριτές μεθοδολογίες για τη μάθηση μεγεθών γειτονιάς ανά δείγμα. Οι Zhang et al. [8] ανέπτυξαν το Correlation Matrix k-NN (CM-kNN) χρησιμοποιώντας αραιή ανακατασκευή. Η μέθοδός τους αναπαριστά κάθε σημείο δοκιμής ως συνδυασμό εκπαιδευτικών σημείων χρησιμοποιώντας συνάρτηση απώλειας ελαχίστων τετραγώνων με ℓ_1-norm regularization, όπου ο αριθμός των σημαντικών συντελεστών ανακατασκευής καθορίζει την προσαρμοστική τιμή k. Αυτή η εξαιρετικά επιδραστική εργασία καθιέρωσε την επιλογή k με βάση τα δεδομένα ως πιο πρακτική από τη cross-validation. Η τεχνική αραιοποίησης που επάγει η ℓ_1-norm προσαρμόζει φυσικά το k με βάση την τοπική πυκνότητα δεδομένων—τα σημεία δοκιμής σε πυκνότερες περιοχές λαμβάνουν μεγαλύτερες τιμές k, ενώ τα απομονωμένα σημεία λαμβάνουν μικρότερες.

Οι Zhang et al. [9] αντιμετώπισαν την υπολογιστική αποδοτικότητα μέσω των μεθόδων kTree και k*Tree, οι οποίες κατασκευάζουν δέντρα απόφασης κατά την εκπαίδευση για να προβλέψουν βέλτιστες τιμές k για νέα δείγματα δοκιμής χωρίς δαπανηρή βελτιστοποίηση ανά ερώ-

τημα. Δημοσιευμένη στο IEEE Transactions on Neural Networks and Learning Systems, αυτή η εργασία προωθεί σημαντικά την πρακτική εφαρμογή των προσαρμοστικών μεθόδων, μειώνοντας την υπολογιστική πολυπλοκότητα από $O(n)$ σε $O(\log n)$ για την πρόβλεψη του k κάθε δείγματος δοκιμής.

2.1.4 Προσεγγίσεις βασισμένες σε Prototypes και Clustering

Μια σημαντική κατηγορία μεθόδων βασίζεται στη χρήση prototypes και clustering για τη δυναμική επιλογή του k. Οι Garcia-Pedrajas, Romero del Castillo και Cerruela-Garcia [10] πρότειναν μια ολοκληρωμένη προσέγγιση όπου ο χώρος αναζήτησης διαιρείται σε υποχώρους, με κάθε υποχώρο να αναπαρίσταται από ένα prototype. Κάθε prototype αντιστοιχίζεται σε μια βέλτιστη τιμή k που προσδιορίζεται μέσω μιας greedy προσέγγισης, δοκιμάζοντας την τοπική απόδοση για $k \in [k_{\min}, k_{\max}]$. Όπως επισημαίζουν οι Papanikolaou et al. [2], αυτή η μέθοδος μπορεί να υιοθετηθεί σε σχεδόν κάθε παραλλαγή k-NN και επέδειξε ανώτερη απόδοση σε 80 datasets για κανονικά προβλήματα και 65 datasets για προβλήματα με ανισορροπημένες κλάσεις. Με 71 αναφορές σε μόλις 4 χρόνια (17.75 αναφορές ανά έτος), η εργασία αυτή αναδείχθηκε ως μία από τις πιο ολοκληρωμένες και επιδραστικές προσεγγίσεις.
Οι Bulut και Amasyali [11] πρότειναν την προσέγγιση One Nearest Cluster (1NC), όπου η τιμή k προσεγγίζεται ως $k \approx M/l$, με M να είναι ο αριθμός των πλησιέστερων δειγμάτων και l ο προκαθορισμένος αριθμός clusters. Η μέθοδος εφαρμόζει clustering στα M πλησιέστερα δείγματα και χρησιμοποιεί majority voting μόνο εντός του πλησιέστερου cluster, επιτυγχάνοντας στατιστικά σημαντική υπεροχή έναντι του 1-NN σε 36 datasets.
Οι Ougiaroglou, Evangelidis και Diamantaras [12] ανέπτυξαν τον Subspace Homogeneity based Dynamic k-NN (shd-k-NN), ο οποίος εφαρμόζει επαναληπτικά την k-means clustering διαδικασία μέχρι όλα τα δημιουργημένα clusters να είναι ομοιογενή. Το βάθος d κάθε ομοιογενούς cluster παρέχει πληροφορίες για την περιοχή όπου βρίσκεται το μη ταξινομημένο δείγμα, και το k υπολογίζεται δυναμικά ως συνάρτηση του d χρησιμοποιώντας διάφορες ευρετικές μεθόδους. Η προσέγγιση αυτή επέδειξε ανταγωνιστική απόδοση σε 14 αρχικά datasets και 19 παραλλαγές τους με τυχαία προσθήκη θορύβου.

2.1.5 Πιθανοτικές προσεγγίσεις

Οι πιθανοτικές προσεγγίσεις αποτελούν μια από τις πλέον διαδεδομένες κατηγορίες μεθόδων για δυναμική επιλογή του k. Οι Ghosh [13, 14] ανέπτυξαν Bayesian προσεγγίσεις για τη βελτιστοποίηση της επιλογής k βασισμένες στην κατανομή δεδομένων, επιδεικνύοντας ότι οι προτεινόμενες μέθοδοι υπερτερούν των συμβατικών τεχνικών cross-validation. Η εργασία του 2007 με 31 αναφορές (2.21 ανά έτος) καθιέρωσε σημαντικά θεωρητικά θεμέλια για την προσαρμοστική επιλογή k με βάση τη συμφωνία ετικετών γειτόνων.
Οι Johansson, Boström και König [15] εισήγαγαν την έννοια των "spheres of confidence" που εκμεταλλεύεται τον εκτιμητή Laplace $P_{ClassA} = \frac{k+1}{N+C}$, όπου k είναι ο αριθμός εκπαιδευτικών δειγμάτων της κλάσης A, N ο συνολικός αριθμός δειγμάτων εντός της σφαίρας, και C ο συνολικός αριθμός κλάσεων. Η μέθοδος κατασκευάζει μια σφαίρα εμπιστοσύνης γύρω από κάθε εκπαιδευτικό δείγμα, με τη διαδικασία να διακόπτεται όταν η τιμή του εκτιμητή Laplace αρχίζει να μειώνεται. Παρά τις μόλις 6 αναφορές, η προσέγγιση αυτή επέδειξε στατιστικά σημαντική υπεροχή έναντι του κλασικού k-NN σε 13 από 18 datasets.

Οι Bhattacharya, Ghosh και Chowdhury [16, 17] πρότειναν την κατασκευή υπερσφαίρας γύρω από κάθε μη ταξινομημένο σημείο για τη σύλληψη της κατανομής των εκπαιδευτικών δειγμάτων γύρω από αυτό, εισάγοντας την έννοια του "hubness weight" την πιθανότητα ένα σημείο να ανήκει στη γειτονιά του συγκεκριμένου σημείου δοκιμής. Η μέθοδος επέδειξε ανώτερη απόδοση σε 15 datasets συγκριτικά με το κλασικό k-NN και άλλες παραλλαγές.

2.1.6 Θεωρητικά θεμέλια και ανάλυση σύγκλισης

Αυστηρές θεωρητικές αναλύσεις παρέχουν τα μαθηματικά υποστηρίγματα για τις προσαρμοστικές μεθόδους k-NN. Οι Cannings, Berrett και Samworth [18] παρήγαγαν ασυμπτωτικές επεκτάσεις για τοπικούς ταξινομητές k-NN όπου το k εξαρτάται από την τοπική οριακή πυκνότητα. Το κύριο αποτέλεσμά τους καθορίζει το βέλτιστο k ως περίπου $k \approx B \times \{n \times \bar{f}(x)\}^{4/(d+4)}$, όπου $\bar{f}(x)$ είναι η τοπική πυκνότητα. Αυτή η προσαρμοστική στην πυκνότητα προσέγγιση επιτυγχάνει ρυθμό σύγκλισης $O(n^{-4/(d+4)})$ υπό ασθενέστερες συνθήκες ροπών από τον τυπικό k-NN. Η εργασία δημοσιεύτηκε στο The Annals of Statistics, ένα από τα κορυφαία θεωρητικά περιοδικά στατιστικής.

Οι Balsubramani et al. [19] παρείχαν θεωρητική ανάλυση για προσαρμοστική επιλογή k με βάση τη συμφωνία ετικετών γειτόνων, εισάγοντας το μέτρο "advantage" μια ποσότητα παρόμοια με το τοπικό περιθώριο. Η εργασία τους αποδεικνύει ότι η πρόβλεψη είναι πιθανώς σωστή όταν τα εκπαιδευτικά σημεία υπερβαίνουν το $O(1/\text{adv}(x))$, καθιερώνοντας συνέπεια υπό ασθενέστερες συνθήκες από τις παραδοσιακές αναλύσεις. Αυτή η προσέγγιση βασισμένη στο advantage παρέχει μια διαφορετική οπτική για το πώς η τοπική δομή των δεδομένων θα πρέπει να καθοδηγεί την επιλογή του k.

Οι Döring, Győrfi και Walk [20] καθιέρωσαν αυστηρά άνω φράγματα για την πιθανότητα πλεονάζοντος σφάλματος του k-NN, δείχνοντας ρυθμό σύγκλισης $O(n^{-2\alpha(1+\alpha)/(d+2\alpha(1+\alpha))})$ υπό συνθήκες περιθωρίου και Lipschitz. Αυτά τα αποτελέσματα ενημερώνουν τον σχεδιασμό προσαρμοστικών παραλλαγών χαρακτηρίζοντας με ακρίβεια την ισορροπία bias-variance. Η εργασία δημοσιεύτηκε στο Journal of Machine Learning Research και αποτελεί σημείο αναφοράς για τη θεωρητική κατανόηση των αλγορίθμων k-NN.

Οι Zhao και Lai [21] καθιέρωσαν θεωρητικά θεμέλια αποδεικνύοντας ότι οι προσαρμοστικοί ταξινομητές k-NN επιτυγχάνουν minimax βέλτιστους ρυθμούς σύγκλισης. Υπό ειδικές υποθέσεις, ο ρυθμός σύγκλισής τους δεν μειώνεται με την αυξανόμενη διαστατικότητα, αντιμετωπίζοντας άμεσα την κατάρα της διαστατικότητας. Η εργασία δημοσιεύτηκε στο AAAI Conference on Artificial Intelligence και αποτελεί σημαντική θεωρητική συνεισφορά στην κατανόηση των ορίων απόδοσης των προσαρμοστικών μεθόδων.

2.1.7 Συγκριτικές αξιολογήσεις και ανασκοπήσεις

Εμπειρικές συγκρίσεις καθιερώνουν τη σχετική απόδοση των προσαρμοστικών μεθόδων. Οι Uddin et al. [22] πραγματοποίησαν ολοκληρωμένη σύγκριση δέκα παραλλαγών k-NN συμπεριλαμβανομένων των Classic, Adaptive, Locally Adaptive, Fuzzy, Mutual και Ensemble προσεγγίσεων σε οκτώ ιατρικά benchmark datasets. Χρησιμοποιώντας τον προτεινόμενο Δείκτη Σχετικής Απόδοσης, διαπίστωσαν ότι ο Hassanat k-NN επέτυχε τη μεγαλύτερη μέση ακρίβεια (83.62%) ενώ η Ensemble Approach k-NN έδειξε την καλύτερη ακρίβεια (82.88%). Οι προσαρμοστικές μέθοδοι υπερτερούσαν σταθερά του κλασικού k-NN κατά 5-10%. Αυτή η μελέτη, δημοσιευμένη

στο Scientific Reports, παρέχει πολύτιμες εμπειρικές πληροφορίες για την επιλογή κατάλληλων μεθόδων k-NN σε ιατρικές εφαρμογές.
Οι Cunningham και Delany [18] παρέχουν ένα ολοκληρωμένο tutorial που καλύπτει μηχανισμούς k-NN, μετρικές απόστασης, υπολογιστικές βελτιστοποιήσεις (kd-trees, ball trees), μείωση διαστάσεων και ομοιότητα χρονοσειρών. Αυτό το άρθρο στο ACM Computing Surveys ανασκοπεί εξελίξεις από το 1951 και συζητά πώς η τοπική προσαρμοστικότητα αντιμετωπίζει την κατάρα της διαστατικότητας. Αποτελεί μια εξαιρετική πηγή για την κατανόηση του ευρύτερου πλαισίου των αλγορίθμων k-NN και των σύγχρονων εξελίξεων.
Η συστηματική βιβλιογραφική ανασκόπηση των Papanikolaou, Evangelidis και Ougiaroglou [2] συγκέντρωσε 28 δημοσιεύσεις που καλύπτουν την περίοδο από το 1986 έως το 2020, ταξινομώντας τις σε έξι κατηγορίες: γενετικούς αλγορίθμους, νευρωνικά δίκτυα, prototypes και clustering, ευρετικές μεθόδους, πιθανοτικές προσεγγίσεις, και άλλες τεχνικές. Η μελέτη τους αποκάλυψε ότι η πλειοψηφία των προσεγγίσεων χρησιμοποιεί πιθανοτικά πλαίσια, αν και αυτές είναι κυρίως οι παλαιότερες εργασίες (διάμεση χρονιά δημοσίευσης 2007). Στις πιο πρόσφατες εργασίες (εντός των τελευταίων πέντε ετών από το 2021), κυριαρχούν οι τεχνικές prototypes και clustering. Η ανάλυση αναφορών έδειξε ότι οι τρεις πιο επιδραστικές εργασίες είναι αυτές των Guo et al. [23] (621 αναφορές, 34.5 ανά έτος), Wang, Neskovic και Cooper [24] (322 αναφορές, 23 ανά έτος), και Song et al. [25] (270 αναφορές, 19.28 ανά έτος).

2.1.8 Εφαρμογές σε ειδικούς τομείς

Οι προσαρμοστικές μέθοδοι k-NN έχουν επιδείξει σημαντικό αντίκτυπο σε διάφορους τομείς εφαρμογών. Οι Li, Shyr και Liu [26] ανέπτυξαν το aKNNO για ομαδοποίηση μονοκυτταρικής και χωρικής μεταγραφωμικής. Η μέθοδός τους επιλέγει προσαρμοστικά το k για κάθε κύτταρο με βάση την τοπική κατανομή αποστάσεων, αντιστοιχίζοντας μικρό k για σπάνια κύτταρα (αφαιρώντας ψευδείς συνδέσεις μεγάλης εμβέλειας) και μεγάλο k για άφθονα κύτταρα (εξισορροπώντας τοπική και καθολική διακύμανση). Σε benchmark με 38 προσομοιωμένα και 20 πραγματικά datasets, το aKNNO υπερτέρησε εξειδικευμένων μεθόδων για την αναγνώριση σπάνιων κυττάρων. Αυτή η εφαρμογή στη βιοπληροφορική δείχνει την ευελιξία των προσαρμοστικών μεθόδων σε υψηλής διάστασης βιολογικά δεδομένα.
Στην κατηγοριοποίηση κειμένων, οι Baoli, Qin και Shiwen [27] ανέπτυξαν μια παραλλαγή k-NN λιγότερο εξαρτώμενη από την τιμή του k, προτείνοντας ότι το k πρέπει να είναι ανάλογο με τον αριθμό εκπαιδευτικών δειγμάτων που ανήκουν στην κατηγορία στην οποία ταξινομείται το δείγμα δοκιμής. Η μέθοδος επέδειξε μειωμένη ευαισθησία στην επιλογή του k και αποτελεσματική αντιμετώπιση της ανισορροπίας κλάσεων, λαμβάνοντας 142 αναφορές (8.35 ανά έτος).

2.1.9 Συμπεράσματα και μελλοντικές κατευθύνσεις

Η ανασκόπηση της βιβλιογραφίας αποκαλύπτει ότι η δυναμική επιλογή του k αποτελεί ένα ενεργό ερευνητικό πεδίο με πολλαπλές προσεγγίσεις που εκμεταλλεύονται διαφορετικές τεχνικές και θεωρητικά πλαίσια. Όπως επισημαίνουν οι Papanikolaou et al. [2], παρά τη μεγάλη ποικιλομορφία των προτεινόμενων μεθόδων, υπάρχει ανάγκη για πιο συστηματικές συγκρίσεις με επαρκή αριθμό datasets και στατιστικούς ελέγχους. Από τις 28 εργασίες που εξέτασαν, μόνο σε 12 περιπτώσεις διεξήχθη τουλάχιστον ένας στατιστικός έλεγχος, ενώ ο διάμεσος αριθ-

μός datasets που χρησιμοποιήθηκαν για αξιολόγηση ήταν μόλις 12. Επιπλέον, οι ερευνητές θα πρέπει να εξετάζουν datasets με θόρυβο, ανισορροπημένες κλάσεις και δεδομένα από πραγματικές εφαρμογές για να αξιολογήσουν ολιστικά την απόδοση κάθε προσέγγισης.
Παρά τις προόδους, παραμένει η πρόκληση της ισορροπίας μεταξύ της επίτευξης υψηλότερων ποσοστών ταξινόμησης (μέσω καλύτερης επιλογής k) και της διατήρησης της σχετικής απλότητας του αλγορίθμου—ένα από τα βασικά πλεονεκτήματα του κλασικού k-NN. Οι μελλοντικές έρευνες θα πρέπει να επικεντρωθούν στην ανάπτυξη μεθόδων που όχι μόνο βελτιώνουν την ακρίβεια αλλά διατηρούν επίσης την υπολογιστική αποδοτικότητα και την ευκολία υλοποίησης, ιδιαίτερα για εφαρμογές σε πραγματικό χρόνο και σε περιβάλλοντα με περιορισμένους υπολογιστικούς πόρους.

Κεφάλαιο 3

Προϋπάρχουσα γνώση

3.1 Ada NN

Ο αλγόριθμος *Adaptive Nearest Neighbors* (AdaNN) [3] αποτελεί επέκταση των κλασικών μεθόδων nearest neighbors, όπου ο αριθμός των γειτόνων δεν είναι σταθερός αλλά προσαρμόζεται δυναμικά για κάθε δείγμα. Όπως αναφέρθηκε στο literature review, το θεμελιώδες άρθρο των Sun και Huang το 2010 εισήγαγε την επαναστατική ιδέα της επιλογής του k ανά δείγμα, αντιμετωπίζοντας τον θεμελιώδη περιορισμό του κλασικού k-NN που χρησιμοποιεί σταθερό μέγεθος γειτονιάς για όλα τα ερωτήματα. Σε αντίθεση με το k-NN, όπου ο αριθμός k είναι κοινός για όλα τα σημεία, στο AdaNN κάθε δείγμα x_i μπορεί να έχει διαφορετικό αριθμό γειτόνων k_i, ανάλογα με την τοπική πυκνότητα και δομή των δεδομένων.
Η βασική ιδέα του AdaNN προέκυψε από την αναγνώριση ότι οι ομοιόμορφες τιμές k αποτυγχάνουν να καταγράψουν τις διακυμάνσεις της τοπικής δομής των δεδομένων. Σε περιοχές υψηλής πυκνότητας απαιτούνται λιγότεροι γείτονες για αξιόπιστη αναπαράσταση της τοπικής δομής, ενώ σε αραιές περιοχές ο αριθμός των γειτόνων πρέπει να αυξάνεται. Έτσι, ο αριθμός k_i προσαρμόζεται ώστε να αντικατοπτρίζει την τοπική γεωμετρία του χώρου χαρακτηριστικών.
Οι Sun και Huang [3] πρότειναν τον προσδιορισμό ενός βέλτιστου k ξεχωριστά για κάθε δείγμα εκπαίδευσης ορισμένο ως τον ελάχιστο αριθμό γειτόνων που απαιτούνται για σωστή ταξινόμηση και στη συνέχεια την κληρονόμηση αυτής της τιμής για δείγματα δοκιμής με βάση τον πλησιέστερο εκπαιδευτικό γείτονά τους. Συγκεκριμένα, για κάθε εκπαιδευτικό δείγμα x_i με ετικέτα y_i, εξετάζουμε διαδοχικά τις τιμές $k = 1, 2, \ldots, k_{\max}$ και επιλέγουμε την πρώτη τιμή για την οποία η πρόβλεψη με k γείτονες δίνει τη σωστή κλάση:

$$k_i = \min\{k : \text{MajorityVote}(\mathcal{N}_k(x_i)) = y_i\}.$$

Αυτή η προσέγγιση, γνωστή ως το κριτήριο του *πρώτου σωστού* k (first correct k), εξασφαλίζει ότι κάθε δείγμα χρησιμοποιεί τον ελάχιστο αναγκαίο αριθμό γειτόνων για σωστή αυτοταξινόμηση, προσαρμοζόμενο στην τοπική πολυπλοκότητα των δεδομένων. Κατά τη φάση της ταξινόμησης ενός νέου δείγματος x_{test}, εντοπίζεται ο πλησιέστερος γείτονας x_{nn} στο εκπαιδευτικό σύνολο και υιοθετείται το αντίστοιχο k_{nn} για την ταξινόμηση.
Το άρθρο των Sun και Huang απέδειξε ότι το ποσοστό σφάλματος του AdaNN βρίσκεται μεταξύ του σφάλματος Bayes και του διπλάσιου σφάλματος Bayes, με την απόδοση στην καλύτερη περίπτωση να πλησιάζει το βέλτιστο σφάλμα Bayes. Τα πειραματικά τους αποτελέσματα σε 15

datasets από το UCI Machine Learning Repository έδειξαν ότι ο AdaNN υπερτερεί των παραδοσιακών k-NN αλγορίθμων σε 11 από τα 15 σύνολα δεδομένων.
Η προσαρμοστικότητα του AdaNN επιτρέπει τη δημιουργία πιο εύρωστων γράφων γειτνίασης, ιδιαίτερα σε σύνολα δεδομένων με ανομοιόμορφη κατανομή. Για τον λόγο αυτό, η μέθοδος χρησιμοποιείται ευρέως σε εφαρμογές όπως ομαδοποίηση, ανίχνευση ανωμαλιών και ημιεπιβλεπόμενη μάθηση. Όπως επισημάνθηκε στην ανασκόπηση της βιβλιογραφίας, η εργασία των Sun και Huang με περίπου 103 αναφορές καθιέρωσε τη θεμελιώδη ιδέα για την προσαρμοστική επιλογή γειτονιάς ανά δείγμα και γέννησε πολυάριθμες επεκτάσεις σε θεωρητική ανάλυση, υπολογιστική βελτιστοποίηση και εφαρμογές σε διάφορους τομείς. Ο Αλγόριθμος 1 παρουσιάζει αναλυτικά τη διαδικασία του AdaNN.

Algorithm 1 AdaNN: Προσαρμοστικός k-NN με First Correct k

Require: Εκπαιδευτικό σύνολο $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, Μέγιστο $k_{\max}$
Ensure: Βέλτιστο k_i για κάθε εκπαιδευτικό δείγμα x_i
Φάση Εκπαίδευσης:
for $i = 1$ **to** n **do**
  Υπολόγισε αποστάσεις: $d_{ij} = \|x_i - x_j\|$ για όλα τα $j \neq i$
  Ταξινόμησε γείτονες: $\text{neighbors}_i = \text{argsort}(\{d_{ij}\}_{j \neq i})$
  $k_i \leftarrow k_{\max}$ ▷ Default τιμή
  for $k = 1$ **to** $k_{\max}$ **do**
    Πάρε τους k πλησιέστερους γείτονες: $\mathcal{N}_k(x_i) = \text{neighbors}_i[1:k]$
    Πρόβλεψη: $\hat{y}_i^{(k)} = \text{MajorityVote}(\{y_j : j \in \mathcal{N}_k(x_i)\})$
    if $\hat{y}_i^{(k)} = y_i$ **then**
      $k_i \leftarrow k$ ▷ Πρώτο σωστό k
      break
    end if
  end for
  Αποθήκευσε k_i για το δείγμα x_i
end for
Φάση Ταξινόμησης:
Require: Νέο δείγμα x_{test}
Βρες πλησιέστερο γείτονα: $x_{\text{nn}} = \arg\min_{x_i \in \mathcal{D}_{\text{train}}} \|x_{\text{test}} - x_i\|$
Υιοθέτησε: $k_{\text{test}} = k_{\text{nn}}$
Βρες τους k_{test} πλησιέστερους γείτονες: $\mathcal{N}_{k_{\text{test}}}(x_{\text{test}})$
return $\hat{y}_{\text{test}} = \text{MajorityVote}(\{y_j : x_j \in \mathcal{N}_{k_{\text{test}}}(x_{\text{test}})\})$

3.2 Mutual k-NN

Ο *Mutual k-Nearest Neighbors* (Mutual k-NN) βασίζεται στην έννοια της αυστηρής αμοιβαίας γειτνίασης[28]. Σε αυτή την προσέγγιση, δύο δείγματα x_i και x_j θεωρούνται αμοιβαίοι γείτονες μόνο αν το καθένα περιλαμβάνεται στο σύνολο των k κοντινότερων γειτόνων του άλλου:

$$x_j \in \text{kNN}(x_i) \quad \text{και} \quad x_i \in \text{kNN}(x_j).$$

Κατά την ταξινόμηση ενός νέου δείγματος x_{test}, ο αλγόριθμος εντοπίζει τους k πλησιέστερους γείτονές του, αλλά **διατηρεί μόνο εκείνους που ικανοποιούν την αμοιβαιότητα**. Δηλαδή, ένας εκπαιδευτικός γείτονας x_j συμμετέχει στην ψηφοφορία ταξινόμησης μόνο αν το x_{test} ανήκει επίσης στους k κοντινότερους γείτονες του x_j (υπολογισμένους στο εκπαιδευτικό σύνολο). Αυτή η λογική **τομής (intersection/AND)** οδηγεί σε αυστηρό φιλτράρισμα και δημιουργία αραιών αλλά εξαιρετικά αξιόπιστων γράφων γειτνίασης, καθώς απορρίπτονται ασυμμετρικές και ενδεχομένως θορυβώδεις συνδέσεις.
Σε περιπτώσεις όπου δεν βρεθούν αμοιβαίοι γείτονες, ο αλγόριθμος επιστρέφει στην κλασική συμπεριφορά k-NN για να αποφευχθεί η αδυναμία ταξινόμησης. Το Mutual k-NN είναι ιδιαίτερα χρήσιμο σε περιβάλλοντα με υψηλό θόρυβο, ανομοιογενή πυκνότητα δεδομένων και σε εφαρμογές ομαδοποίησης όπου η ποιότητα των συνδέσεων είναι κρισιμότερη από την ποσότητα. Ο Αλγόριθμος 2 περιγράφει τη διαδικασία του Mutual k-NN.

Algorithm 2 Mutual k-NN: k-NN με Αμοιβαίους Γείτονες

Require: Εκπαιδευτικό σύνολο $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, Παράμετρος k
Ensure: Πρόβλεψη $\hat{y}_{\text{test}}$ για νέο δείγμα
Φάση Ταξινόμησης:
Require: Νέο δείγμα x_{test}
Βρες τους k πλησιέστερους γείτονες: $\mathcal{N}_k(x_{\text{test}})$
$\mathcal{M} \leftarrow \emptyset$ ▷ Σύνολο αμοιβαίων γειτόνων
for each $x_j \in \mathcal{N}_k(x_{\text{test}})$ **do**
    Υπολόγισε τους k πλησιέστερους γείτονες του x_j: $\mathcal{N}_k(x_j)$
    if $\mathcal{N}_k(x_{\text{test}}) \cap \mathcal{N}_k(x_j) \neq \emptyset$ **then**
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{x_j\}$ ▷ Αμοιβαία σχέση
    end if
end for
if $\mathcal{M} = \emptyset$ **then**
    return MajorityVote$(\{y_j : x_j \in \mathcal{N}_k(x_{\text{test}})\})$ ▷ Fallback
else
    return $\hat{y}_{\text{test}} =$ MajorityVote$(\{y_j : x_j \in \mathcal{M}\})$
end if

3.3 Symmetric k-NN

Ο *Symmetric k-Nearest Neighbors* (Symmetric k-NN) αποτελεί μια εναλλακτική προσέγγιση στη συμμετρική γειτνίαση που, αντί να φιλτράρει γείτονες όπως ο Mutual k-NN, τους **εμπλουτίζει** με επιπλέον σημεία. Η βασική ιδέα είναι η δημιουργία ενός εκτεταμένου δικτύου γειτνίασης που περιλαμβάνει τόσο τους άμεσους γείτονες όσο και τους γείτονες των γειτόνων[29].
Συγκεκριμένα, κατά την ταξινόμηση ενός νέου δείγματος x_{test}, ο αλγόριθμος:

1. Εντοπίζει τους k πλησιέστερους γείτονες του x_{test}
2. Για κάθε γείτονα x_j, υπολογίζει τους δικούς του k πλησιέστερους γείτονες

3. Συνενώνει όλα αυτά τα σημεία (λογική ένωσης - union/OR)

4. Χρησιμοποιεί το εμπλουτισμένο σύνολο για την ψηφοφορία ταξινόμησης

Αυτή η προσέγγιση δημιουργεί ένα πυκνότερο και πιο πληροφοριακά πλούσιο δίκτυο γειτνίασης, καθώς λαμβάνει υπόψη την ευρύτερη τοπολογία των δεδομένων. Σε αντίθεση με τον Mutual k-NN που είναι περιοριστικός, ο Symmetric k-NN είναι συμπεριληπτικός και μπορεί να οδηγήσει σε πιο εύρωστες αποφάσεις ταξινόμησης, ειδικά σε περιπτώσεις όπου το δείγμα δοκιμής βρίσκεται κοντά στα σύνορα μεταξύ κλάσεων. Ο Αλγόριθμος 3 απεικονίζει τη διαδικασία του Symmetric k-NN.

Algorithm 3 Symmetric k-NN: k-NN με Συμμετρικούς Γείτονες

Require: Εκπαιδευτικό σύνολο $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, Παράμετρος k
Ensure: Πρόβλεψη $\hat{y}_{\text{test}}$ για νέο δείγμα
Φάση Ταξινόμησης:
Require: Νέο δείγμα x_{test}
Βρες τους k πλησιέστερους γείτονες: $\mathcal{N}_k(x_{\text{test}})$
$\mathcal{S} \leftarrow \mathcal{N}_k(x_{\text{test}})$ ▷ Αρχικοποίηση
for each $x_j \in \mathcal{N}_k(x_{\text{test}})$ **do**
    Υπολόγισε τους k πλησιέστερους γείτονες του x_j: $\mathcal{N}_k(x_j)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{N}_k(x_j)$ ▷ Ένωση γειτόνων
end for
return $\hat{y}_{\text{test}} = \text{MajorityVote}(\{y_j : x_j \in \mathcal{S}\})$

Κεφάλαιο 4

Δικιές μας προτάσεις

4.1 Ada Mutual k-NN

Η πρώτη προτεινόμενη μέθοδος, η **Ada Mutual k-NN**, αποτελεί έναν υβριδικό αλγόριθμο που συνδυάζει τα πλεονεκτήματα της προσαρμοστικότητας του AdaNN με την αξιοπιστία των αμοιβαίων γειτονικών σχέσεων του Mutual k-NN. Η βασική ιδέα πίσω από αυτή την προσέγγιση είναι ότι η απλή προσαρμογή του αριθμού γειτόνων για κάθε δείγμα, όπως γίνεται στον κλασικό AdaNN, μπορεί να οδηγήσει σε ασύμμετρες γειτονικές σχέσεις που ενδέχεται να εισάγουν θόρυβο στην ταξινόμηση. Για παράδειγμα, ένα δείγμα x_i μπορεί να θεωρεί το x_j ως γείτονα με βάση το βέλτιστο k_i που έχει προσδιοριστεί για αυτό, ενώ το x_j με το δικό του βέλτιστο k_j μπορεί να μην ανταποδίδει αυτή τη σχέση γειτνίασης. Αυτές οι ασυμμετρίες μπορούν να είναι προβληματικές, ιδιαίτερα σε περιοχές των δεδομένων όπου η πυκνότητα μεταβάλλεται απότομα ή όπου υπάρχουν outliers. Στην Ada Mutual k-NN, για κάθε δείγμα προσδιορίζουμε αρχικά το βέλτιστο k_i χρησιμοποιώντας τη μεθοδολογία του AdaNN, δηλαδή βρίσκουμε τον ελάχιστο αριθμό γειτόνων που απαιτείται για τη σωστή ταξινόμηση κάθε εκπαιδευτικού δείγματος. Στη συνέχεια, κατά τη φάση της ταξινόμησης ενός νέου δείγματος x_{test}, εντοπίζουμε τον πλησιέστερο γείτονά του x_{nn} στο εκπαιδευτικό σύνολο και υιοθετούμε το βέλτιστο k_{nn} του. Ωστόσο, προτού προχωρήσουμε στην τελική ταξινόμηση, επιβάλλουμε την επιπλέον συνθήκη της αμοιβαιότητας: για κάθε υποψήφιο γείτονα x_j από τους k_{nn} πλησιέστερους στο x_{test}, υπολογίζουμε τους k_j κοντινότερους γείτονες του x_j στο εκπαιδευτικό σύνολο. Το x_j διατηρείται ως αμοιβαίος γείτονας μόνο αν κάποιο σημείο από τη λίστα των k_{nn} πλησιέστερων γειτόνων του x_{test} ανήκει επίσης στη λίστα των k_j γειτόνων του x_j. Αυτός ο επιπλέον έλεγχος αμοιβαιότητας φιλτράρει τις ασταθείς ή τυχαίες γειτονικές σχέσεις και διασφαλίζει ότι μόνο οι “ισχυρές” και αξιόπιστες συνδέσεις λαμβάνονται υπόψη στην τελική απόφαση ταξινόμησης. Σε περίπτωση που δεν βρεθούν αμοιβαίοι γείτονες, ο αλγόριθμος επιστρέφει στην κλασική συμπεριφορά του AdaNN για να αποφευχθεί η αδυναμία ταξινόμησης.

Algorithm 4 Ada Mutual k-NN: Προσαρμοστικός k-NN με Αμοιβαίους Γείτονες

Require: Εκπαιδευτικό σύνολο $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, Μέγιστο $k_{\max}$
Ensure: Βέλτιστο k_i για κάθε εκπαιδευτικό δείγμα x_i
Φάση Εκπαίδευσης: (Ίδια με AdaNN - Αλγόριθμος 1)
for $i = 1$ **to** n **do**
    Υπολόγισε k_i όπως στον AdaNN (γραμμές 2-12 του Αλγορίθμου 1)
end for
Φάση Ταξινόμησης:
Require: Νέο δείγμα x_{test}
Βρες πλησιέστερο γείτονα: $x_{\text{nn}} = \arg\min_{x_i \in \mathcal{D}_{\text{train}}} \|x_{\text{test}} - x_i\|$
Υιοθέτησε: $k_{\text{test}} = k_{\text{nn}}$
Βρες τους k_{test} πλησιέστερους γείτονες: $\mathcal{N}_{k_{\text{test}}}(x_{\text{test}})$
$\mathcal{M} \leftarrow \emptyset$ ▷ Σύνολο αμοιβαίων γειτόνων
for each $x_j \in \mathcal{N}_{k_{\text{test}}}(x_{\text{test}})$ **do**
    Βρες τους k_j πλησιέστερους γείτονες του x_j: $\mathcal{N}_{k_j}(x_j)$
    if $\mathcal{N}_{k_{\text{test}}}(x_{\text{test}}) \cap \mathcal{N}_{k_j}(x_j) \neq \emptyset$ **then**
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{x_j\}$ ▷ Προσθήκη αμοιβαίου γείτονα
    end if
end for
if $\mathcal{M} = \emptyset$ **then**
    return MajorityVote$(\{y_j : x_j \in \mathcal{N}_{k_{\text{test}}}(x_{\text{test}})\})$ ▷ Fallback σε AdaNN
else
    return $\hat{y}_{\text{test}} =$ MajorityVote$(\{y_j : x_j \in \mathcal{M}\})$
end if

4.2 Ada Symmetric k-NN

Η δεύτερη προτεινόμενη μέθοδος, η **Ada Symmetric k-NN**, αποτελεί μια εναλλακτική προσέγγιση που επεκτείνει την ιδέα της συμμετρίας στις γειτονικές σχέσεις με διαφορετικό τρόπο από την Ada Mutual k-NN. Ενώ η Ada Mutual k-NN εστιάζει στην αμοιβαιότητα μεταξύ συγκεκριμένων ζευγών σημείων, η Ada Symmetric k-NN υιοθετεί μια πιο ευρεία και συμπεριληπτική προσέγγιση που στοχεύει στη δημιουργία ενός εμπλουτισμένου συνόλου γειτόνων. Συγκεκριμένα, κατά την ταξινόμηση ενός νέου δείγματος x_{test}, η Ada Symmetric k-NN ακολουθεί την ίδια αρχική διαδικασία με την Ada Mutual k-NN: εντοπίζει τον πλησιέστερο γείτονα x_{nn} στο εκπαιδευτικό σύνολο και υιοθετεί το βέλτιστο k_{nn} του. Στη συνέχεια, όμως, αντί να φιλτράρει τους γείτονες με βάση την αυστηρή αμοιβαιότητα, εφαρμόζει μια λογική ένωσης (union): ξεκινά με το σύνολο των k_{nn} πλησιέστερων γειτόνων του x_{test}, και για κάθε έναν από αυτούς τους γείτονες x_j, υπολογίζει τους k_j δικούς του κοντινότερους γείτονες στο εκπαιδευτικό σύνολο. Όλα αυτά τα σημεία προστίθενται στο τελικό σύνολο γειτόνων που θα χρησιμοποιηθεί για την ψηφοφορία ταξινόμησης. Με αυτόν τον τρόπο, η Ada Symmetric k-NN διατηρεί όχι μόνο τους άμεσους γείτονες του δείγματος δοκιμής αλλά και τους γείτονες των γειτόνων, δημιουργώντας ένα πυκνότερο και πιο πληροφοριακά πλούσιο δίκτυο γειτνίασης.
Η κύρια διαφορά και το πλεονέκτημα της Ada Symmetric k-NN έναντι της Ada Mutual k-NN έγκειται στο ότι είναι λιγότερο περιοριστική και διατηρεί μεγαλύτερο όγκο πληροφορίας από

τη γειτονική δομή των δεδομένων. Αντί να απορρίπτει γείτονες που δεν ικανοποιούν την αυστηρή συνθήκη αμοιβαιότητας, η Ada Symmetric k-NN εμπλουτίζει το σύνολο των γειτόνων με πρόσθετα σημεία που βρίσκονται στη γειτονιά των αρχικών γειτόνων. Η μέθοδος μοιάζει εννοιολογικά με την κατασκευή ενός γράφου δεύτερης τάξης, όπου λαμβάνονται υπόψη όχι μόνο οι άμεσες αλλά και οι έμμεσες συνδέσεις μεταξύ των σημείων. Ο πλουσιότερος αυτός χώρος γειτνίασης μπορεί να οδηγήσει σε πιο εύρωστες αποφάσεις ταξινόμησης, ειδικά σε περιπτώσεις όπου το δείγμα δοκιμής βρίσκεται κοντά στα σύνορα μεταξύ κλάσεων ή σε περιοχές με περίπλοκη τοπολογία.

Αναμένεται ότι η Ada Symmetric k-NN θα παρουσιάσει διαφορετική συμπεριφορά από την Ada Mutual k-NN: ενώ η δεύτερη θα είναι πιο συντηρητική και επιλεκτική στους γείτονες που χρησιμοποιεί, η πρώτη θα είναι πιο συμπεριληπτική και θα αξιοποιεί ευρύτερο πλαίσιο πληροφορίας. Η πειραματική αξιολόγηση και σύγκριση των δύο μεθόδων σε διάφορα datasets θα αποκαλύψει ποια προσέγγιση (η επιλεκτική αμοιβαιότητα ή η συμπεριληπτική ένωση) είναι πιο αποτελεσματική υπό διαφορετικές συνθήκες και χαρακτηριστικά δεδομένων, παρέχοντας σημαντικές πληροφορίες για την επιλογή της κατάλληλης μεθόδου ανάλογα με το πρόβλημα.

Algorithm 5 Ada Symmetric k-NN: Προσαρμοστικός k-NN με Συμμετρικούς Γείτονες

Require: Εκπαιδευτικό σύνολο $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, Μέγιστο $k_{\max}$
Ensure: Βέλτιστο k_i για κάθε εκπαιδευτικό δείγμα x_i
Φάση Εκπαίδευσης: (Ίδια με AdaNN - Αλγόριθμος 1)
for $i = 1$ **to** n **do**
    Υπολόγισε k_i όπως στον AdaNN (γραμμές 2-12 του Αλγορίθμου 1)
end for
Φάση Ταξινόμησης:
Require: Νέο δείγμα x_{test}
Βρες πλησιέστερο γείτονα: $x_{\text{nn}} = \arg\min_{x_i \in \mathcal{D}_{\text{train}}} \|x_{\text{test}} - x_i\|$
Υιοθέτησε: $k_{\text{test}} = k_{\text{nn}}$
Βρες τους k_{test} πλησιέστερους γείτονες: $\mathcal{N}_{k_{\text{test}}}(x_{\text{test}})$
$\mathcal{S} \leftarrow \mathcal{N}_{k_{\text{test}}}(x_{\text{test}})$ ▷ Αρχικοποίηση με άμεσους γείτονες
for each $x_j \in \mathcal{N}_{k_{\text{test}}}(x_{\text{test}})$ **do**
    Βρες τους k_j πλησιέστερους γείτονες του x_j: $\mathcal{N}_{k_j}(x_j)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{N}_{k_j}(x_j)$ ▷ Ένωση (union) γειτόνων
end for
return $\hat{y}_{\text{test}} = \text{MajorityVote}(\{y_j : x_j \in \mathcal{S}\})$

4.3 Ada Last Stable k-NN

Η τρίτη προτεινόμενη μέθοδος, η **Ada Last Stable k-NN**, εισάγει μια εναλλακτική φιλοσοφία στον προσδιορισμό του βέλτιστου αριθμού γειτόνων για κάθε εκπαιδευτικό δείγμα, στοχεύοντας στην επίτευξη μεγαλύτερης σταθερότητας και αξιοπιστίας στις προβλέψεις. Ενώ οι κλασικές προσεγγίσεις όπως ο AdaNN επιλέγουν το *πρώτο* σωστό k για κάθε δείγμα—δηλαδή τον ελάχιστο αριθμό γειτόνων που απαιτείται για σωστή ταξινόμηση—η Ada Last Stable k-NN υιοθετεί μια πιο συντηρητική και σταθερή στρατηγική: αναζητά το *τελευταίο* σταθερό k που διατηρεί σταθερή και σωστή πρόβλεψη για ένα συγκεκριμένο παράθυρο τιμών.

Η κεντρική ιδέα βασίζεται στην παρατήρηση ότι η απλή επιλογή του πρώτου σωστού k μπορεί να οδηγήσει σε ασταθείς αποφάσεις, ιδιαίτερα όταν η σωστή ταξινόμηση επιτυγχάνεται τυχαία ή σε οριακές περιπτώσεις με μικρό αριθμό γειτόνων. Για παράδειγμα, ένα δείγμα μπορεί να ταξινομηθεί σωστά με $k = 2$, αλλά η πρόβλεψη να αλλάξει λανθασμένα με $k = 3$, να επανέλθει σωστή με $k = 4$, και να παραμείνει σταθερά σωστή για $k = 5, 6, 7$. Σε μια τέτοια περίπτωση, η επιλογή $k = 2$ μπορεί να είναι επικίνδυνη καθώς η απόφαση είναι εύθραυστη και ευαίσθητη σε μικρές μεταβολές. Αντίθετα, η επιλογή ενός k από τη σταθερή περιοχή ($k = 5, 6, 7$) παρέχει μεγαλύτερη εμπιστοσύνη στην αξιοπιστία της πρόβλεψης.

Συγκεκριμένα, κατά τη φάση της εκπαίδευσης, για κάθε εκπαιδευτικό δείγμα x_i, η Ada Last Stable k-NN υπολογίζει προβλέψεις για όλες τις τιμές k από 1 έως k_{max}, δημιουργώντας μια ακολουθία προβλέψεων $\{\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \ldots, \hat{y}_i^{(k_{\text{max}})}\}$. Στη συνέχεια, εφαρμόζει ένα κυλιόμενο παράθυρο σταθερότητας μεγέθους w (συνήθως $w = 3$) και εντοπίζει όλα τα διαστήματα όπου: (α) όλες οι προβλέψεις εντός του παραθύρου είναι ίδιες, και (β) η πρόβλεψη συμφωνεί με την πραγματική ετικέτα y_i. Από όλα αυτά τα "σταθερά παράθυρα", η μέθοδος επιλέγει το *τελευταίο*—δηλαδή αυτό που αντιστοιχεί στη μεγαλύτερη τιμή k—και ορίζει ως βέλτιστο k_i το τέλος αυτού του παραθύρου. Με αυτόν τον τρόπο, εξασφαλίζεται ότι το επιλεγμένο k_i δεν αντιπροσωπεύει μια μεμονωμένη σωστή πρόβλεψη, αλλά βρίσκεται σε μια ευρύτερη περιοχή σταθερότητας.

Η φιλοσοφία της Ada Last Stable k-NN αντικατοπτρίζει μια ισορροπία μεταξύ της αποφυγής υπερπροσαρμογής (overfitting) και της αποφυγής υποπροσαρμογής (underfitting). Επιλέγοντας ένα μεγαλύτερο k από μια σταθερή περιοχή, η μέθοδος ενσωματώνει πληροφορία από ευρύτερη γειτονιά, μειώνοντας την ευαισθησία σε θόρυβο και outliers, ενώ ταυτόχρονα διατηρεί την ακρίβεια της πρόβλεψης. Επιπλέον, η απαίτηση για συνεχόμενη σταθερότητα σε ένα παράθυρο w τιμών εξασφαλίζει ότι η επιλογή δεν είναι τυχαία αλλά αντανακλά μια πραγματική δομική ιδιότητα των δεδομένων στη συγκεκριμένη περιοχή του χώρου χαρακτηριστικών.

Σε περιπτώσεις όπου δεν εντοπίζεται κανένα σταθερό παράθυρο για ένα συγκεκριμένο δείγμα—κάτι που μπορεί να συμβεί σε περιοχές με υψηλή πολυπλοκότητα ή έντονη επικάλυψη κλάσεων—η μέθοδος υιοθετεί ως προεπιλεγμένη τιμή το k_{max}, εξασφαλίζοντας έτσι ότι χρησιμοποιείται μια επαρκώς μεγάλη γειτονιά για την απόφαση ταξινόμησης. Κατά τη φάση της ταξινόμησης ενός νέου δείγματος, η Ada Last Stable k-NN ακολουθεί την ίδια διαδικασία με τον κλασικό AdaNN: εντοπίζει τον πλησιέστερο γείτονα στο εκπαιδευτικό σύνολο και υιοθετεί το αντίστοιχο βέλτιστο k που έχει προσδιοριστεί με τη μέθοδο του τελευταίου σταθερού παραθύρου.

Η Ada Last Stable k-NN αναμένεται να παρουσιάσει διαφορετική συμπεριφορά σε σύγκριση με τον κλασικό AdaNN και τις άλλες προτεινόμενες μεθόδους. Ενώ ο AdaNN τείνει να επιλέγει μικρότερες τιμές k (πιο τοπικές αποφάσεις), η Ada Last Stable k-NN αναμένεται να επιλέγει μεγαλύτερες τιμές k που βρίσκονται σε σταθερές περιοχές, οδηγώντας σε πιο εύρωστες και γενικεύσιμες προβλέψεις. Αυτό μπορεί να είναι ιδιαίτερα επωφελές σε datasets με θόρυβο, outliers, ή περιοχές με ασαφή όρια μεταξύ κλάσεων. Η πειραματική αξιολόγηση θα αποκαλύψει κατά πόσον η αυξημένη σταθερότητα που επιτυγχάνεται με αυτή την προσέγγιση μεταφράζεται σε βελτιωμένη ακρίβεια ταξινόμησης και γενίκευση σε νέα δεδομένα.

Algorithm 6 Ada Last Stable k-NN: Προσαρμοστικός k-NN με Τελευταίο Σταθερό k

Require: Εκπαιδευτικό σύνολο $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, Μέγιστο k_{max}, Μέγεθος παραθύρου w
Ensure: Βέλτιστο k_i για κάθε εκπαιδευτικό δείγμα x_i

Φάση Εκπαίδευσης:
for $i = 1$ **to** n **do**
    Υπολόγισε αποστάσεις: $d_{ij} = \|x_i - x_j\|$ για όλα τα $j \neq i$
    Ταξινόμησε γείτονες: neighbors$_i$ = argsort$(\{d_{ij}\}_{j \neq i})$
    $k_i \leftarrow k_{\text{max}}$ ▷ Default τιμή
    Υπολόγισε προβλέψεις για όλα τα k:
    for $k = 1$ **to** k_{max} **do**
        $\mathcal{N}_k(x_i)$ = neighbors$_i[1:k]$
        pred$[k]$ = MajorityVote$(\{y_j : j \in \mathcal{N}_k(x_i)\})$
    end for
    Βρες τελευταίο σταθερό παράθυρο:
    for $k = 1$ **to** $k_{\text{max}} - w + 1$ **do**
        window = {pred$[k]$, pred$[k+1]$, ..., pred$[k+w-1]$}
        if όλα τα στοιχεία του window είναι ίδια **AND** pred$[k] = y_i$ **then**
            $k_i \leftarrow k + w - 1$ ▷ Τέλος σταθερού παραθύρου
        end if
    end for
    Αποθήκευσε k_i για το δείγμα x_i
end for
Φάση Ταξινόμησης: (Ίδια με AdaNN - Αλγόριθμος 1, γραμμές 14-18)

Κεφάλαιο 5

Κώδικας εργασίας

5.1 Python

Η Python είναι μία σύγχρονη γλώσσα προγραμματισμού υψηλού επιπέδου, η οποία διακρίνεται για την απλότητα της σύνταξής της και την ευκολία χρήσης. Υποστηρίζει πολλαπλά προγραμματιστικά παραδείγματα, όπως αντικειμενοστραφή και συναρτησιακό προγραμματισμό, και διαθέτει πλούσιο οικοσύστημα βιβλιοθηκών. Λόγω της εκτεταμένης υποστήριξης για ανάλυση δεδομένων, μηχανική μάθηση και επιστημονικούς υπολογισμούς, η Python χρησιμοποιείται ευρέως στην υλοποίηση και αξιολόγηση αλγορίθμων.

5.1.1 Βιβλιοθήκες για την Υλοποίηση του k-NN

NumPy

Η βιβλιοθήκη `numpy` αποτελεί τη θεμελιώδη βιβλιοθήκη για επιστημονικούς υπολογισμούς στην Python. Παρέχει υποστήριξη για πολυδιάστατους πίνακες και πίνακες υψηλής απόδοσης, καθώς και ένα ευρύ φάσμα μαθηματικών συναρτήσεων για την επεξεργασία τους. Στο πλαίσιο του k-NN, η NumPy χρησιμοποιείται για την αποθήκευση και επεξεργασία των διανυσμάτων χαρακτηριστικών, καθώς και για τον υπολογισμό των αποστάσεων μεταξύ των σημείων δεδομένων με αποδοτικό τρόπο.

Collections - Counter

Η κλάση `Counter` από τη βιβλιοθήκη `collections` αποτελεί μία ειδική δομή δεδομένων για την καταμέτρηση αντικειμένων. Στον αλγόριθμο k-NN, χρησιμοποιείται για την καταμέτρηση των ετικετών των k πλησιέστερων γειτόνων και την εύρεση της πιο συχνής κατηγορίας, η οποία αποτελεί την τελική πρόβλεψη του αλγορίθμου.

Scikit-learn Metrics

Η βιβλιοθήκη `sklearn.metrics` παρέχει ένα ολοκληρωμένο σύνολο μετρικών αξιολόγησης για μοντέλα μηχανικής μάθησης. Οι συναρτήσεις `accuracy_score`, `precision_score`, `recall_score` και `f1_score` υπολογίζουν τις αντίστοιχες μετρικές απόδοσης του ταξινομητή. Συγκεκριμένα, η ακρίβεια (accuracy) μετρά το ποσοστό των σωστών προβλέψεων, η

ακρίβεια κατηγορίας (precision) το ποσοστό των θετικών προβλέψεων που είναι πραγματικά θετικές, η ανάκληση (recall) το ποσοστό των θετικών περιπτώσεων που ανιχνεύθηκαν σωστά, και η μετρική F1 το αρμονικό μέσο της ακρίβειας και της ανάκλησης. Η συνάρτηση `classification_report` παράγει μια λεπτομερή αναφορά με όλες τις παραπάνω μετρικές για κάθε κατηγορία.

Pandas

Η βιβλιοθήκη `pandas` αποτελεί το βασικό εργαλείο για τη διαχείριση και ανάλυση δεδομένων στην Python. Παρέχει δομές δεδομένων όπως το DataFrame, οι οποίες διευκολύνουν την οργάνωση, επεξεργασία και ανάλυση των δεδομένων. Στα πειράματα k-NN, η Pandas χρησιμοποιείται για τη φόρτωση των συνόλων δεδομένων, την προεπεξεργασία τους, καθώς και για την οργάνωση και παρουσίαση των αποτελεσμάτων των πειραμάτων σε μορφή πινάκων.

Βιβλιοθήκες Συστήματος και Χρόνου

Οι βιβλιοθήκες `os`, `time`, `math` και `datetime` παρέχουν βασικές λειτουργίες για την αλληλεπίδραση με το λειτουργικό σύστημα, τη μέτρηση του χρόνου εκτέλεσης, μαθηματικούς υπολογισμούς και τη διαχείριση ημερομηνιών και ωρών αντίστοιχα. Η βιβλιοθήκη `os` χρησιμοποιείται για την πλοήγηση στο σύστημα αρχείων και τη διαχείριση αρχείων δεδομένων. Η `time` επιτρέπει την καταγραφή του χρόνου εκτέλεσης των αλγορίθμων, πληροφορία κρίσιμη για την αξιολόγηση της υπολογιστικής αποδοτικότητας. Η `math` παρέχει μαθηματικές συναρτήσεις όπως η τετραγωνική ρίζα για τον υπολογισμό αποστάσεων, ενώ η `datetime` χρησιμοποιείται για τη χρονοσήμανση των πειραμάτων και την οργάνωση των αποτελεσμάτων.

Concurrent Futures

Η βιβλιοθήκη `concurrent.futures` παρέχει ένα υψηλού επιπέδου περιβάλλον για την ασύγχρονη εκτέλεση κώδικα. Συγκεκριμένα, η κλάση `ProcessPoolExecutor` δημιουργεί μια δεξαμενή εργατικών διεργασιών (worker processes) που επιτρέπει την παράλληλη εκτέλεση εργασιών σε πολλαπλούς επεξεργαστές. Αυτό είναι ιδιαίτερα χρήσιμο για τον αλγόριθμο k-NN, καθώς οι υπολογισμοί αποστάσεων και οι προβλέψεις για διαφορετικά σημεία δεδομένων μπορούν να εκτελεστούν ανεξάρτητα και παράλληλα, επιταχύνοντας σημαντικά τη συνολική διαδικασία αξιολόγησης, ειδικά σε μεγάλα σύνολα δεδομένων. Η συνάρτηση `as_completed` επιστρέφει τα αποτελέσματα των παράλληλων εργασιών καθώς ολοκληρώνονται, επιτρέποντας την αποδοτική διαχείριση και συλλογή των αποτελεσμάτων.

5.2 Οργάνωση κώδικα

Ο κώδικας υλοποιεί τις ακόλουθες λειτουργίες:

- Φόρτωση δεδομένων από αρχεία csv με διαχωρισμό χαρακτηριστικών και ετικετών
- Δημιουργία συνθετικών datasets με εισαγωγή παραμετροποιημένου ποσοστού θορύβου (π.χ. 10%, 30%, 50%) για την αξιολόγηση της ανθεκτικότητας των αλγορίθμων

- Υπολογισμός του βέλτιστου (adaptive) k για όλα τα δείγματα του training dataset με παραμετροποίηση του $k_{\max}$ (επιλογές: 1, 5, 9, 20, 50, $\sqrt{n/2}$, κτλ.) για διερεύνηση της επίδρασης του ανώτατου ορίου γειτόνων
- Υπολογισμός παραλλαγής του adaptive k με χρήση της μεθόδου "last k" που ενσωματώνει παράθυρο σταθερότητας (stability window) παραμετροποιημένου μεγέθους (π.χ. 3, 5, 7) για πιο εύρωστη εκτίμηση του βέλτιστου k
- Υπολογισμός του βέλτιστου global k για κάθε dataset και κάθε μέθοδο μέσω k-fold cross validation (π.χ. 5-fold ή 10-fold) για την εύρεση της βέλτιστης παραμέτρου που μεγιστοποιεί την ακρίβεια ταξινόμησης
- Εκτέλεση αλγορίθμων k-NN με καταγραφή χρόνου και υπολογισμό μετρικών αξιολόγησης
- Παράλληλη επεξεργασία τριών παραλλαγών k-NN για κάθε σύνολο δεδομένων
- Παράλληλη επεξεργασία πολλαπλών συνόλων δεδομένων από φάκελο
- Εμφάνιση αποτελεσμάτων κατά την εκτέλεση
- Αποθήκευση αποτελεσμάτων σε αρχείο Excel με χρονοσήμανση
- Δημιουργία συγκεντρωτικού φύλλου με όλες τις μετρικές
- Δημιουργία ξεχωριστών φύλλων με αναλυτικές αναφορές ταξινόμησης για κάθε πείραμα

5.3 Υλοποίηση παραλλαγών Knn

Listing 5.1: Python example

```
import numpy as np
from collections import Counter
from sklearn.metrics import accuracy_score


def euclidean_distances(X, x):
    return np.linalg.norm(X - x, axis=1)


def min_max_normalize(X):
    denom = X.max(axis=0) - X.min(axis=0)
    denom[denom == 0] = 1
    return (X - X.min(axis=0)) / denom


class CustomKNN:
```

```
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y
        return self

    def predict(self, X_test):
        return np.array([self._predict(x) for x in X_test])

    def _predict(self, x):
        distances = euclidean_distances(self.X_train, x)
        k_indices = np.argsort(distances)[:self.k]
        labels = self.y_train[k_indices]
        return Counter(labels).most_common(1)[0][0]

    def score(self, X, y):
        return accuracy_score(y, self.predict(X))

    def get_params(self, deep=True):
        return {"k": self.k}

    def set_params(self, **params):
        for key, value in params.items():
            setattr(self, key, value)
        return self


class MutualKNN(CustomKNN):
    """
␣␣␣␣Mutual␣k-NN:␣Keeps␣only␣neighbors␣where␣the␣relationship␣is␣mutual.
␣␣␣␣A␣training␣point␣is␣a␣mutual␣neighbor␣if␣the␣test␣point␣would␣be
␣␣␣␣among␣its␣k␣nearest␣neighbors.
␣␣␣␣"""
    def _predict(self, x):
        distances = euclidean_distances(self.X_train, x)
        k_indices = np.argsort(distances)[:self.k]

        mutual_labels = []

        for idx in k_indices:
            back_distances = euclidean_distances(self.X_train, self.
               X_train[idx])
            back_sorted = np.argsort(back_distances)

            # Exclude self (distance zero)
            k_back = back_sorted[1:self.k + 1]
```

```
            k_th_distance = back_distances[k_back[-1]]

            if distances[idx] <= k_th_distance:
                mutual_labels.append(self.y_train[idx])

        if mutual_labels:
            return Counter(mutual_labels).most_common(1)[0][0]
        else:
            return super()._predict(x)


class SymmetricKNN(CustomKNN):
    """
␣␣␣␣Symmetric␣k-NN␣(2-hop␣/␣union␣k-NN):
␣␣␣␣Uses␣the␣union␣of␣the␣k␣nearest␣neighbors␣of␣the␣test␣point
␣␣␣␣and␣the␣k␣nearest␣neighbors␣of␣each␣of␣those␣neighbors.
␣␣␣␣"""
    def _predict(self, x):
        distances = euclidean_distances(self.X_train, x)
        k_indices = np.argsort(distances)[:self.k]

        all_neighbors = set(k_indices)

        for idx in k_indices:
            back_distances = euclidean_distances(self.X_train, self.
               X_train[idx])
            k_back = np.argsort(back_distances)[1:self.k + 1]
            all_neighbors.update(k_back)

        labels = self.y_train[list(all_neighbors)]
        return Counter(labels).most_common(1)[0][0]
```

5.4 Υλοποίηση

5.4.1 Υπολογισμός First Correct k (AdaNN)

Ο παρακάτω κώδικας υπολογίζει το πρώτο σωστό k για κάθε εκπαιδευτικό δείγμα:

Listing 5.2: Υπολογισμός First Correct k

```
def compute_first_correct_k(X, y, model_name, max_k):
    n = len(y)
    first_correct_k = np.full(n, max_k)  # Default: max_k
    dist_matrix = compute_distance_matrix(X)
    sorted_neighbors = np.argsort(dist_matrix, axis=1)[:, 1:]

    for i in range(n):
```

```
        neighbors = sorted_neighbors[i]
        for k in range(1, max_k + 1):
            # Πρόβλεψημε  k γείτονες
            pred = predict_label(y[neighbors], k, model_name)
            if pred == y[i]:
                first_correct_k[i] = k
                break  # Πρώτοσωστό  k

    return first_correct_k
```

5.4.2 Υπολογισμός Last Stable k

Ο αλγόριθμος Last Stable k βρίσκει το τελευταίο σταθερό παράθυρο προβλέψεων:

Listing 5.3: Υπολογισμός Last Stable k

```
def compute_adaptive_k_last_stable(X, y, model_name, max_k, window=3):
    n = len(y)
    adaptive_k = np.full(n, max_k)
    dist_matrix = compute_distance_matrix(X)
    sorted_neighbors = np.argsort(dist_matrix, axis=1)[:, 1:]

    for i in range(n):
        neighbors = sorted_neighbors[i]
        preds = []

        # Υπολογισμόςπροβλέψεωνγιαόλατα     k
        for k in range(1, max_k + 1):
            pred = predict_label(y[neighbors], k, model_name)
            preds.append(pred)

        # Εύρεσητελευταίουσταθερούπαραθύρου
        last_k = max_k
        for k in range(1, max_k - window + 2):
            window_preds = preds[k-1:k-1+window]
            # Έλεγχος: όλαίδια  AND σωστά
            if (all(p == window_preds[0] for p in window_preds) and
                window_preds[0] == y[i]):
                last_k = k + window - 1  # Τέλοςπαραθύρου

        adaptive_k[i] = last_k

    return adaptive_k
```

5.4.3 Ταξινόμηση με Adaptive k

Κατά την ταξινόμηση, χρησιμοποιείται το adaptive k του πλησιέστερου γείτονα:

Listing 5.4: Ταξινόμηση με Adaptive k

```
def classify_with_adaptive_k(X_train, y_train, k_per_row, x_test):
    # Εύρεσηπλησιέστερουγείτονα
    distances = np.linalg.norm(X_train - x_test, axis=1)
    nearest_idx = np.argmin(distances)

    # Υιοθέτησητου  k τουπλησιέστερουγείτονα
    k_adaptive = k_per_row[nearest_idx]

    # Ταξινόμησημετο   adaptive k
    model = CustomKNN(k=k_adaptive)
    model.fit(X_train, y_train)
    prediction = model.predict([x_test])[0]

    return prediction, k_adaptive
```

5.4.4 Cross-Validation για Best k

Εύρεση βέλτιστου k με 5-fold cross-validation:

Listing 5.5: Cross-Validation για Best k

```
def find_best_k_cv(X_train, y_train, k_range, n_folds=5):
    kf = KFold(n_splits=n_folds, shuffle=True, random_state=42)
    k_scores = {k: [] for k in k_range}

    for k in k_range:
        for train_idx, val_idx in kf.split(X_train):
            X_tr, X_val = X_train[train_idx], X_train[val_idx]
            y_tr, y_val = y_train[train_idx], y_train[val_idx]

            model = CustomKNN(k=k)
            model.fit(X_tr, y_tr)
            predictions = model.predict(X_val)
            accuracy = accuracy_score(y_val, predictions)
            k_scores[k].append(accuracy)

    # Επιλογή k μεμέγιστομέσο   accuracy
    mean_scores = {k: np.mean(scores) for k, scores in k_scores.items()
       }
    best_k = max(mean_scores, key=mean_scores.get)

    return best_k, mean_scores[best_k]
```

5.4.5 Friedman Test για Στατιστική Σύγκριση

Στατιστική σύγκριση μεθόδων με Friedman test:

Listing 5.6: Friedman Test

```
from scipy import stats

def perform_friedman_test(accuracy_matrix):
    # Πίνακας: γραμμές=datasets, στήλες=methods
    n_datasets, n_methods = accuracy_matrix.shape

    # Friedman test
    statistic, p_value = stats.friedmanchisquare(
        *[accuracy_matrix[:, i] for i in range(n_methods)]
    )

    # Υπολογισμός ranks χαμηλότερο( rank = καλύτερημέθοδος )
    ranks = np.zeros_like(accuracy_matrix)
    for i in range(n_datasets):
        ranks[i, :] = stats.rankdata(-accuracy_matrix[i, :])

    average_ranks = ranks.mean(axis=0)

    return {
        'statistic': statistic,
        'p_value': p_value,
        'average_ranks': average_ranks,
        'significant': p_value < 0.05
    }
```

5.4.6 Wilcoxon Signed-Rank Test

Pairwise σύγκριση μεθόδων με Wilcoxon test:

Listing 5.7: Wilcoxon Pairwise Test

```
from itertools import combinations

def pairwise_wilcoxon(accuracy_matrix, method_names):
    results = []

    for i, j in combinations(range(len(method_names)), 2):
        method1, method2 = method_names[i], method_names[j]

        # Wilcoxon signed-rank test
        statistic, p_value = stats.wilcoxon(
            accuracy_matrix[:, i],
            accuracy_matrix[:, j],
            alternative='two-sided'
        )

```

```
16          mean_diff = accuracy_matrix[:, i].mean() - accuracy_matrix[:, j
                ].mean()
17          better = method1 if mean_diff > 0 else method2
18
19          results.append({
20              'Method_1': method1,
21              'Method_2': method2,
22              'p_value': p_value,
23              'Better': better,
24              'Significant': p_value < 0.05
25          })
26
27      return pd.DataFrame(results).sort_values('p_value')
```

Κεφάλαιο 6

Πειραματική Μελέτη

6.1 Σύνολα Δεδομένων

Στην πειραματική μελέτη χρησιμοποιήθηκαν δεδομένα από το UCI Machine Learning Repository [30] και το KEEL Repository [31].

6.1.1 Magic Gamma Telescope (MGT)

Το σύνολο δεδομένων Magic Gamma Telescope [32] περιέχει δεδομένα από προσομοιώσεις τηλεσκοπίου Cherenkov υψηλής ενέργειας. Στόχος του dataset είναι η διάκριση μεταξύ σημάτων που προκαλούνται από πρωτογενή σωματίδια gamma (σήμα) και εκείνων που προκαλούνται από αδρόνια (background noise). Το dataset αποτελείται από 19,020 παρατηρήσεις με 10 χαρακτηριστικά που περιγράφουν τις ιδιότητες των καταγεγραμμένων ατμοσφαιρικών καταιονισμών και 2 κλάσεις (gamma και hadron). Χρησιμοποιείται ευρέως σε προβλήματα ταξινόμησης και αξιολόγησης αλγορίθμων machine learning στον τομέα της αστροφυσικής.

6.1.2 Pen-Based Recognition of Handwritten Digits (PD)

Το Pen-Digits dataset [33] περιέχει δεδομένα χειρόγραφων ψηφίων που συλλέχθηκαν με τη χρήση ηλεκτρονικού στυλό σε tablet. Κάθε ψηφίο (0-9) αναπαρίσταται από μια χρονική ακολουθία συντεταγμένων (x, y) που καταγράφηκαν κατά τη διάρκεια της γραφής. Το dataset αποτελείται από 10,992 δείγματα με 16 χαρακτηριστικά που αντιπροσωπεύουν τις χωρικές συντεταγμένες σε διάφορα χρονικά σημεία και 10 κλάσεις (ψηφία 0-9). Πρόκειται για ένα κλασικό benchmark dataset για προβλήματα αναγνώρισης προτύπων και ταξινόμησης πολλαπλών κλάσεων.

6.1.3 Landsat Satellite (LS)

Το Landsat Satellite dataset [34] προέρχεται από δορυφορικές εικόνες πολυφασματικής ανίχνευσης της επιφάνειας της Γης. Περιλαμβάνει δεδομένα που αφορούν διάφορους τύπους εδαφικής κάλυψης όπως βλάστηση, έδαφος, νερό και αστικές περιοχές. Το dataset αποτελείται από 6,435 pixel values με 36 χαρακτηριστικά που αντιπροσωπεύουν φασματικές πληροφορίες από διαφορετικά κανάλια του δορυφόρου και 6 κλάσεις (red soil, cotton crop, grey soil, damp grey

soil, soil with vegetation stubble, very damp grey soil). Χρησιμοποιείται συχνά σε εφαρμογές τηλεπισκόπησης, κατηγοριοποίησης τοπίου και περιβαλλοντικής παρακολούθησης.

6.1.4 Phoneme (PH)

Το Phoneme dataset [35] περιέχει δεδομένα ακουστικής ανάλυσης για την αναγνώριση φωνημάτων από συνεχή ομιλία. Κάθε δείγμα αντιστοιχεί σε ένα τμήμα ομιλίας που έχει κατηγοριοποιηθεί ως ένα από δύο διαφορετικά φωνήματα (nasal και oral). Το dataset περιλαμβάνει 5,404 παρατηρήσεις με 5 χαρακτηριστικά που προέρχονται από ανάλυση φασματικών ιδιοτήτων του σήματος ομιλίας και 2 κλάσεις. Αποτελεί σημαντικό εργαλείο για την αξιολόγηση αλγορίθμων αναγνώρισης ομιλίας και επεξεργασίας φυσικής γλώσσας.

6.1.5 Banana (BN)

Το Banana dataset [36] είναι ένα συνθετικό σύνολο δεδομένων που δημιουργήθηκε για benchmarking αλγορίθμων ταξινόμησης. Οι παρατηρήσεις κατανέμονται σε δύο κλάσεις που σχηματίζουν μοτίβα σε σχήμα μπανάνας σε διδιάστατο χώρο χαρακτηριστικών. Το dataset αποτελείται από 5,300 δείγματα με μόλις 2 χαρακτηριστικά και 2 κλάσεις, καθιστώντας το ιδανικό για οπτικοποίηση και κατανόηση της συμπεριφοράς ταξινομητών σε μη γραμμικά προβλήματα. Χρησιμοποιείται ευρέως για τη δοκιμή μεθόδων που χειρίζονται μη γραμμικά διαχωρίσιμες κλάσεις.

6.1.6 Yeast (YS)

Το Yeast dataset [37] περιέχει δεδομένα πρωτεϊνικής ακολουθίας για την πρόβλεψη της υποκυτταρικής τοποθεσίας πρωτεϊνών στον ζυμομύκητα Saccharomyces cerevisiae. Κάθε παράδειγμα αντιπροσωπεύει μια πρωτεΐνη με χαρακτηριστικά που περιγράφουν φυσικοχημικές ιδιότητες και αλληλουχίες σημάτων. Το dataset αποτελείται από 1,484 παρατηρήσεις με 8 χαρακτηριστικά και 10 κλάσεις που αντιστοιχούν σε διαφορετικές υποκυτταρικές τοποθεσίες. Είναι ιδιαίτερα χρήσιμο για προβλήματα πολυκλασικής ταξινόμησης στη βιοπληροφορική.

6.1.7 Pima Indians Diabetes (PM)

Το Pima Indians Diabetes dataset [38] περιέχει ιατρικά δεδομένα από γυναίκες της φυλής Pima Indians για την πρόβλεψη της εμφάνισης διαβήτη. Τα χαρακτηριστικά περιλαμβάνουν κλινικές μετρήσεις όπως επίπεδα γλυκόζης στο αίμα, δείκτη μάζας σώματος (BMI), ηλικία και ιστορικό εγκυμοσύνης. Το dataset αποτελείται από 768 παρατηρήσεις με 8 αριθμητικά χαρακτηριστικά και μια δυαδική μεταβλητή εξόδου που δείχνει την παρουσία ή απουσία διαβήτη. Χρησιμοποιείται ευρέως σε ιατρική πληροφορική και προγνωστική ανάλυση.

6.1.8 Texture (TXR)

Το Texture dataset [39] περιέχει χαρακτηριστικά που εξάγονται από εικόνες υφής διαφόρων επιφανειών και υλικών. Τα χαρακτηριστικά υπολογίζονται με τη χρήση στατιστικών μεθόδων ανάλυσης υφής όπως οι πίνακες συνεμφάνισης (co-occurrence matrices) και wavelet transforms.

Το dataset αποτελείται από 5,500 δείγματα με 40 χαρακτηριστικά που αντιπροσωπεύουν διάφορες ιδιότητες υφής όπως ομοιογένεια, αντίθεση και ενέργεια. Χρησιμοποιείται σε εφαρμογές όρασης υπολογιστών, αναγνώρισης προτύπων και ανάλυσης εικόνας.

6.1.9 Shuttle (SH)

Το Shuttle dataset [40] προέρχεται από δεδομένα διαστημικού λεωφορείου (Space Shuttle) και περιέχει πληροφορίες για την κατάσταση διαφόρων συστημάτων κατά τη διάρκεια πτήσεων. Στόχος είναι η κατηγοριοποίηση της θέσης ραδιοσυχνοτήτων (radiator positions) σε επτά διαφορετικές κλάσεις. Το dataset αποτελείται από 58,000 παρατηρήσεις (43,500 training, 14,500 test) με 9 αριθμητικά χαρακτηριστικά που περιγράφουν διάφορες τεχνικές παραμέτρους και 7 κλάσεις. Χαρακτηρίζεται από σημαντική ανισορροπία κλάσεων (περίπου 80% ανήκει στην κλάση 1), καθιστώντας το κατάλληλο για δοκιμή αλγορίθμων που χειρίζονται imbalanced datasets.

6.1.10 E. coli (ECL)

Το E. coli dataset [41] περιέχει δεδομένα για την πρόβλεψη της τοποθεσίας πρωτεϊνών στο βακτήριο Escherichia coli. Τα χαρακτηριστικά προέρχονται από ανάλυση αμινοξικών ακολουθιών και περιλαμβάνουν πληροφορίες για σηματοδοτικές αλληλουχίες και περιεχόμενο αμινοξέων. Το dataset αποτελείται από 336 παρατηρήσεις με 7 αριθμητικά χαρακτηριστικά και 8 κλάσεις που αντιστοιχούν σε διαφορετικές κυτταρικές τοποθεσίες (cp, im, pp, imU, om, omL, imL, imS). Χρησιμοποιείται κυρίως σε εφαρμογές βιοπληροφορικής και μοριακής βιολογίας.

6.1.11 Iris (IRIS)

Το Iris dataset [42] είναι ένα από τα πιο διάσημα και παλαιότερα datasets στη στατιστική και το machine learning, εισηχθέν από τον Ronald Fisher το 1936. Περιέχει μετρήσεις από 150 δείγματα λουλουδιών ίριδας τριών διαφορετικών ειδών (Setosa, Versicolor, Virginica). Κάθε δείγμα περιγράφεται από 4 χαρακτηριστικά: μήκος και πλάτος σέπαλων (sepal length, sepal width) και μήκος και πλάτος πετάλων (petal length, petal width) και 3 κλάσεις (50 δείγματα ανά κλάση). Λόγω της απλότητας και της καλής διαχωρισιμότητας των κλάσεων, χρησιμοποιείται ευρέως ως εισαγωγικό παράδειγμα σε μαθήματα machine learning.

6.1.12 Twonorm (TN)

Το Twonorm dataset [43] είναι ένα συνθετικό benchmark dataset που δημιουργήθηκε για την αξιολόγηση αλγορίθμων ταξινόμησης. Τα δεδομένα προέρχονται από δύο πολυδιάστατες κανονικές κατανομές (Gaussian distributions) με διαφορετικούς μέσους όρους αλλά ίδιους πίνακες συνδιακύμανσης. Το dataset αποτελείται από 7,400 παρατηρήσεις με 20 χαρακτηριστικά και 2 κλάσεις. Χρησιμοποιείται για τη δοκιμή της ικανότητας των ταξινομητών να χειρίζονται υψηλοδιάστατα δεδομένα με επικαλυπτόμενες κατανομές κλάσεων.

6.1.13 Balance Scale (BL)

Το Balance Scale dataset [44] περιέχει δεδομένα που προσομοιώνουν ένα πείραμα ψυχολογίας για την κατανόηση της ισορροπίας από παιδιά. Κάθε παράδειγμα περιγράφει μια ζυγαριά με βάρη σε διαφορετικές αποστάσεις από το κέντρο ισορροπίας. Το dataset αποτελείται από 625 παρατηρήσεις με 4 χαρακτηριστικά (αριστερό βάρος, αριστερή απόσταση, δεξί βάρος, δεξιά απόσταση) και 3 κλάσεις: αριστερά (288 δείγματα), δεξιά (288 δείγματα), ισορροπημένο (49 δείγματα). Χρησιμοποιείται για προβλήματα πολυκλασικής ταξινόμησης και συμβολικής μάθησης.

6.1.14 Letter Recognition (LIR)

Το Letter Recognition dataset [45] περιέχει χαρακτηριστικά που εξάγονται από εικόνες των 26 κεφαλαίων γραμμάτων του αγγλικού αλφαβήτου. Τα χαρακτηριστικά υπολογίζονται με βάση στατιστικές μετρήσεις από pixel-based representations των γραμμάτων, όπως οριζόντια και κατακόρυφη θέση, πλάτος, ύψος και διάφορες στατιστικές κατανομής. Το dataset αποτελείται από 20,000 παρατηρήσεις με 16 αριθμητικά χαρακτηριστικά και 26 κλάσεις (γράμματα A-Z). Χρησιμοποιείται ευρέως σε προβλήματα οπτικής αναγνώρισης χαρακτήρων (OCR) και ταξινόμησης πολλαπλών κλάσεων.

6.1.15 Συγκεντρωτικός Πίνακας

Στον Πίνακα 6.1 παρουσιάζεται συγκεντρωτικά ο αριθμός παρατηρήσεων, χαρακτηριστικών και κλάσεων για κάθε dataset.

Πίνακας 6.1: Συγκεντρωτικά χαρακτηριστικά των συνόλων δεδομένων

Dataset	**Παρατηρήσεις**	**Χαρακτηριστικά**	**Κλάσεις**
Magic Gamma Telescope (MGT)	19,020	10	2
Pen-Digits (PD)	10,992	16	10
Landsat Satellite (LS)	6,435	36	6
Phoneme (PH)	5,404	5	2
Banana (BN)	5,300	2	2
Yeast (YS)	1,484	8	10
Pima Indians Diabetes (PM)	768	8	2
Texture (TXR)	5,500	40	11
Shuttle (SH)	58,000	9	7
E. coli (ECL)	336	7	8
Iris (IRIS)	150	4	3
Twonorm (TN)	7,400	20	2
Balance Scale (BL)	625	4	3
Letter Recognition (LIR)	20,000	16	26

6.2 Εγκαθίδρυση πειραμάτων

6.2.1 Περιβάλλον υλοποίησης

Η υλοποίηση και εκτέλεση των πειραμάτων πραγματοποιήθηκε σε υπολογιστικό σύστημα με τις ακόλουθες προδιαγραφές:

- **Επεξεργαστής:** AMD Ryzen 7 7840HS με ενσωματωμένα γραφικά Radeon 780M (8 πυρήνες, 16 νήματα) στα 3.80 GHz
- **Μνήμη RAM:** 16.0 GB (15.3 GB διαθέσιμα)
- **Λειτουργικό Σύστημα:** Ubuntu 22.04.5 LTS εκτελούμενο σε Windows Subsystem for Linux 2 (WSL2)
- **Kernel:** Linux 6.6.87.2-microsoft-standard-WSL2
- **Γλώσσα Προγραμματισμού:** Python 3.x
- **Περιβάλλον Εκτέλεσης:** Windows Terminal με Bash shell (έκδοση 5.1.16)

Η χρήση του WSL2 επιτρέπει την εκτέλεση πλήρους Linux περιβάλλοντος με εγγενή υποστήριξη για επιστημονικές βιβλιοθήκες Python, διατηρώντας παράλληλα την ευελιξία του Windows συστήματος. Ο επεξεργαστής AMD Ryzen 7 7840HS βασίζεται στην αρχιτεκτονική Zen 4 και παρέχει υψηλή υπολογιστική απόδοση για εφαρμογές μηχανικής μάθησης, ενώ τα 16GB RAM εξασφαλίζουν επαρκή μνήμη για την επεξεργασία των datasets που χρησιμοποιούνται στην παρούσα εργασία.

6.2.2 Μετρικές αξιολόγησης

Για την αξιολόγηση της απόδοσης των αλγορίθμων k-NN και των παραλλαγών τους, χρησιμοποιήθηκε ένα σύνολο πέντε βασικών μετρικών που καλύπτουν τόσο την ταχύτητα εκτέλεσης όσο και την ποιότητα ταξινόμησης. Οι μετρικές αυτές παρουσιάζονται αναλυτικά παρακάτω:

Χρόνος εκτέλεσης (CPU time)

Ο χρόνος εκτέλεσης μετρά τον συνολικό χρόνο σε δευτερόλεπτα που απαιτείται για την ταξινόμηση όλων των δειγμάτων του συνόλου δοκιμής. Η μέτρηση περιλαμβάνει τον χρόνο υπολογισμού των αποστάσεων, την αναζήτηση των k πλησιέστερων γειτόνων και την εφαρμογή της τεχνικής majority voting για την τελική πρόβλεψη. Αποτελεί κρίσιμο δείκτη της υπολογιστικής αποδοτικότητας κάθε μεθόδου, ιδιαίτερα για εφαρμογές σε πραγματικό χρόνο ή για μεγάλα σύνολα δεδομένων.

Ακρίβεια (Accuracy)

Η ακρίβεια ορίζεται ως το ποσοστό των σωστά ταξινομημένων δειγμάτων επί του συνολικού αριθμού δειγμάτων στο σύνολο δοκιμής:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

όπου TP (True Positives) είναι τα αληθώς θετικά, TN (True Negatives) τα αληθώς αρνητικά, FP (False Positives) τα ψευδώς θετικά και FN (False Negatives) τα ψευδώς αρνητικά. Για προβλήματα πολλαπλών κλάσεων, η ακρίβεια υπολογίζεται ως ο λόγος του συνόλου των σωστών προβλέψεων προς το σύνολο των προβλέψεων. Παρά την ευρεία χρήση της, η ακρίβεια μπορεί να είναι παραπλανητική σε περιπτώσεις ανισορροπημένων κλάσεων, όπου μία κλάση κυριαρχεί αριθμητικά.

Μέση Ακρίβεια ανά Κλάση (Average Precision)

Η μέση ακρίβεια υπολογίζει την ακρίβεια (precision) για κάθε κλάση ξεχωριστά και στη συνέχεια λαμβάνει τον μέσο όρο των τιμών αυτών. Η ακρίβεια για μία κλάση c ορίζεται ως:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}$$

και η μέση ακρίβεια ως:

$$\text{Average Precision} = \frac{1}{C} \sum_{c=1}^{C} \text{Precision}_c$$

όπου C είναι ο αριθμός των κλάσεων. Αυτή η μετρική απαντά στο ερώτημα: ”Από όλα τα δείγματα που προβλέφθηκαν ως κλάση c, πόσα πράγματι ανήκουν σε αυτή την κλάση;” Είναι ιδιαίτερα χρήσιμη όταν το κόστος των ψευδώς θετικών είναι υψηλό.

Μέση Ανάκληση ανά Κλάση (Average Recall)

Η μέση ανάκληση υπολογίζει την ανάκληση (recall) για κάθε κλάση ξεχωριστά και στη συνέχεια λαμβάνει τον μέσο όρο. Η ανάκληση για μία κλάση c ορίζεται ως:

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}$$

και η μέση ανάκληση ως:

$$\text{Average Recall} = \frac{1}{C} \sum_{c=1}^{C} \text{Recall}_c$$

Αυτή η μετρική απαντά στο ερώτημα: ”Από όλα τα δείγματα που πράγματι ανήκουν στην κλάση c, πόσα εντοπίστηκαν σωστά από τον ταξινομητή;” Είναι ιδιαίτερα σημαντική όταν το κόστος των ψευδώς αρνητικών είναι υψηλό, όπως σε ιατρικές διαγνώσεις.

F1-Score

Το F1-Score αποτελεί τον αρμονικό μέσο της ακρίβειας και της ανάκλησης, παρέχοντας μία ισορροπημένη μετρική που λαμβάνει υπόψη και τις δύο διαστάσεις της απόδοσης. Για κάθε κλάση c υπολογίζεται ως:

$$\text{F1-Score}_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

και το τελικό F1-Score ως ο μέσος όρος των F1-Score όλων των κλάσεων:

$$\text{F1-Score} = \frac{1}{C}\sum_{c=1}^{C}\text{F1-Score}_c$$

Η χρήση του αρμονικού μέσου εξασφαλίζει ότι το F1-Score είναι χαμηλό όταν είτε η ακρίβεια είτε η ανάκληση είναι χαμηλή, καθιστώντας το πιο αυστηρή μετρική από τον αριθμητικό μέσο. Το F1-Score είναι ιδιαίτερα χρήσιμο σε περιπτώσεις ανισορροπημένων κλάσεων, καθώς δεν επηρεάζεται από την πλειοψηφία των αληθώς αρνητικών όπως η ακρίβεια.
Στην παρούσα εργασία, όλες οι μετρικές υπολογίζονται για κάθε dataset ξεχωριστά, επιτρέποντας μια ολοκληρωμένη αξιολόγηση της απόδοσης κάθε αλγορίθμου υπό διαφορετικές συνθήκες και χαρακτηριστικά δεδομένων. Η συνδυασμένη χρήση αυτών των μετρικών παρέχει μια πλήρη εικόνα τόσο της υπολογιστικής αποδοτικότητας όσο και της ποιότητας ταξινόμησης.

6.3 Μετρήσεις πειραμάτων

Μετρήσεις στα 14 datasets χωρίς θόρυβο

Πίνακας 6.2: Σύγκριση μεθόδων KNN με k=1

Dataset	Μετρική	Conventional KNN k=1	Mutual k=1	Symmetric k=1
	CPU time (sec)	5.2873	13.1770	11.2330
	Accuracy	0.801262	0.801262	0.799159
mgt	Avg Precision	0.786831	0.786831	0.784884
	Avg Recall	0.770850	0.770850	0.767560
	F1-Score	0.777291	0.777291	0.774408
	CPU time (sec)	93.1948	209.3057	181.8371
	Accuracy	0.999569	0.999569	0.999224
sh	Avg Precision	0.913087	0.913087	0.917345
	Avg Recall	0.924060	0.924060	0.910652
	F1-Score	0.894544	0.894544	0.889374
	CPU time (sec)	0.6418	1.7829	1.3159
	Accuracy	0.955405	0.955405	0.943243
tn	Avg Precision	0.955416	0.955416	0.943301
	Avg Recall	0.955409	0.955409	0.943236
	F1-Score	0.955405	0.955405	0.943241
	CPU time (sec)	0.0005	0.0006	0.0005
	Accuracy	0.900000	0.900000	0.900000
iris	Avg Precision	0.902357	0.902357	0.902357
	Avg Recall	0.900000	0.900000	0.900000
	F1-Score	0.899749	0.899749	0.899749
	CPU time (sec)	1.1076	3.0511	2.3027
	Accuracy	0.899767	0.899767	0.884227
ls	Avg Precision	0.885542	0.885542	0.869746
	Avg Recall	0.890340	0.890340	0.873959
	F1-Score	0.887777	0.887777	0.871683
	CPU time (sec)	0.1239	0.2965	0.2479
	Accuracy	0.872642	0.872642	0.871698
bn	Avg Precision	0.871411	0.871411	0.870681
	Avg Recall	0.872222	0.872222	0.870681

Dataset	Μετρική	Conventional KNN k=1	Mutual k=1	Symmetric k=1
	F1-Score	0.871778	0.871778	0.870681
	CPU time (sec)	0.0039	0.0048	0.0058
	Accuracy	0.800000	0.800000	0.768000
bl	Avg Precision	0.566508	0.566508	0.571053
	Avg Recall	0.574990	0.574990	0.550468
	F1-Score	0.568127	0.568127	0.553540
	CPU time (sec)	0.0135	0.0417	0.0285
	Accuracy	0.493243	0.493243	0.496622
ys	Avg Precision	0.525945	0.525945	0.504282
	Avg Recall	0.486404	0.486404	0.460526
	F1-Score	0.498764	0.498764	0.476040
	CPU time (sec)	0.2075	0.3923	0.3698
	Accuracy	0.887037	0.887037	0.870370
ph	Avg Precision	0.872800	0.872800	0.851633
	Avg Recall	0.851317	0.851317	0.831369
	F1-Score	0.860921	0.860921	0.840401
	CPU time (sec)	1.7224	4.1208	3.0168
	Accuracy	0.990446	0.990446	0.988171
pd	Avg Precision	0.990554	0.990554	0.988423
	Avg Recall	0.990642	0.990642	0.988287
	F1-Score	0.990577	0.990577	0.988323
	CPU time (sec)	0.0012	0.0019	0.0015
	Accuracy	0.835821	0.835821	0.880597
ecl	Avg Precision	0.694444	0.694444	0.761265
	Avg Recall	0.692361	0.692361	0.714683
	F1-Score	0.692489	0.692489	0.733760
	CPU time (sec)	0.6718	1.5753	1.1370
	Accuracy	0.987273	0.987273	0.986364
txr	Avg Precision	0.987395	0.987395	0.986576
	Avg Recall	0.987759	0.987759	0.986947
	F1-Score	0.987522	0.987522	0.986707
	CPU time (sec)	0.0030	0.0078	0.0060
	Accuracy	0.771242	0.771242	0.712418
pm	Avg Precision	0.747475	0.747475	0.681373
	Avg Recall	0.749623	0.749623	0.678019
	F1-Score	0.748509	0.748509	0.679551
	CPU time (sec)	19.7163	32.6909	28.3432
	Accuracy	0.956500	0.956500	0.941750
lir	Avg Precision	0.955648	0.955648	0.940889
	Avg Recall	0.955951	0.955951	0.941158
	F1-Score	0.955671	0.955671	0.940806

Τα αποτελέσματα της σύγκρισης των τριών μεθόδων KNN για $k = 1$ παρουσιάζονται στον Πίνακα 6.2. Παρατηρείται ότι ο Conventional KNN και ο Mutual KNN επιτυγχάνουν ταυτόσημη ακρίβεια σε όλα τα datasets, ενώ ο Symmetric KNN παρουσιάζει χαμηλότερη απόδοση στα περισσότερα datasets, με τις μεγαλύτερες αποκλίσεις να εμφανίζονται στα pm (5.9% χαμηλότερη ακρίβεια), bl (3.2% χαμηλότερη), και ph (1.7% χαμηλότερη). Ωστόσο, ο Symmetric KNN υπερτερεί στο dataset ecl με 4.5% υψηλότερη ακρίβεια σε σχέση με τις άλλες δύο μεθόδους. Από πλευράς υπολογιστικής απόδοσης, ο Conventional KNN είναι σταθερά ο ταχύτερος, με τον Mutual KNN να απαιτεί περίπου 2-3 φορές περισσότερο χρόνο εκτέλεσης και τον Symmetric KNN να βρίσκεται μεταξύ των δύο.

Πίνακας 6.3: Σύγκριση μεθόδων KNN με k=5

Dataset	Μετρική	Conventional KNN k=5	Mutual k=5	Symmetric k=5
	CPU time (sec)	4.9275	26.6583	26.8864
	Accuracy	0.829653	0.819926	0.824395
mgt				

Dataset	Μετρική	Conventional KNN k=5	Mutual k=5	Symmetric k=5
	Avg Precision	0.828891	0.808857	0.834946
	Avg Recall	0.791214	0.790471	0.775854
	F1-Score	0.804009	0.797854	0.792470
	CPU time (sec)	94.1694	479.4691	478.4194
	Accuracy	0.999052	0.999483	0.998793
sh	Avg Precision	0.928084	0.908893	0.708037
	Avg Recall	0.870440	0.923980	0.654682
	F1-Score	0.871956	0.892375	0.678435
	CPU time (sec)	0.6537	4.0986	3.7972
	Accuracy	0.977027	0.962162	0.976351
tn	Avg Precision	0.977029	0.962172	0.976377
	Avg Recall	0.977029	0.962166	0.976347
	F1-Score	0.977027	0.962162	0.976351
	CPU time (sec)	0.0003	0.0011	0.0013
	Accuracy	0.933333	0.933333	0.933333
iris	Avg Precision	0.944444	0.944444	0.944444
	Avg Recall	0.933333	0.933333	0.933333
	F1-Score	0.932660	0.932660	0.932660
	CPU time (sec)	1.1856	9.7732	9.8031
	Accuracy	0.915307	0.909868	0.898213
ls	Avg Precision	0.906396	0.895603	0.891915
	Avg Recall	0.901035	0.893707	0.880229
	F1-Score	0.903288	0.894330	0.885160
	CPU time (sec)	0.1390	0.7682	0.7440
	Accuracy	0.898113	0.895283	0.900943
bn	Avg Precision	0.898473	0.896115	0.901247
	Avg Recall	0.895956	0.892682	0.898893
	F1-Score	0.897004	0.894036	0.899885
	CPU time (sec)	0.0028	0.0108	0.0122
	Accuracy	0.864000	0.864000	0.888000
bl	Avg Precision	0.606820	0.604167	0.597509
	Avg Recall	0.621795	0.621795	0.641026
	F1-Score	0.611854	0.611111	0.617249
	CPU time (sec)	0.0216	0.0890	0.0794
	Accuracy	0.570946	0.516892	0.550676
ys	Avg Precision	0.601766	0.499143	0.476510
	Avg Recall	0.550951	0.474214	0.448065
	F1-Score	0.567289	0.482816	0.456275
	CPU time (sec)	0.1674	1.1037	1.1205
	Accuracy	0.869444	0.887037	0.849074
ph	Avg Precision	0.851326	0.868599	0.826788
	Avg Recall	0.828913	0.857609	0.801836
	F1-Score	0.838782	0.862791	0.812498
	CPU time (sec)	1.7273	9.9841	9.7104
	Accuracy	0.990901	0.993631	0.986806
pd	Avg Precision	0.991083	0.993645	0.987315
	Avg Recall	0.991262	0.993785	0.987498
	F1-Score	0.991131	0.993698	0.987311
	CPU time (sec)	0.0008	0.0037	0.0042
	Accuracy	0.895522	0.865672	0.895522
ecl	Avg Precision	0.899348	0.755626	0.935948
	Avg Recall	0.911690	0.741468	0.883565

Dataset	Μετρική	Conventional KNN k=5	Mutual k=5	Symmetric k=5
	F1-Score	0.903977	0.746028	0.900737
	CPU time (sec)	0.5823	3.4448	3.6929
	Accuracy	0.980909	0.988182	0.975455
txr	Avg Precision	0.981112	0.988520	0.975890
	Avg Recall	0.981807	0.988502	0.976462
	F1-Score	0.981381	0.988466	0.975958
	CPU time (sec)	0.0031	0.0187	0.0186
	Accuracy	0.784314	0.771242	0.771242
pm	Avg Precision	0.767306	0.747430	0.774923
	Avg Recall	0.741887	0.745189	0.700849
	F1-Score	0.750950	0.746269	0.714932
	CPU time (sec)	19.7324	57.5837	58.4261
	Accuracy	0.954000	0.956000	0.928250
lir	Avg Precision	0.953244	0.955294	0.928039
	Avg Recall	0.953978	0.955444	0.927594
	F1-Score	0.953258	0.955162	0.927258

Τα αποτελέσματα για $k = 5$ παρουσιάζονται στον Πίνακα 6.3. Ο Conventional KNN επιτυγχάνει την υψηλότερη ακρίβεια στα περισσότερα datasets (mgt, tn, ls, ys, pm, lir), ενώ ο Mutual KNN υπερτερεί στα pd (0.27% υψηλότερη ακρίβεια), ph (1.76% υψηλότερη), sh (0.04% υψηλότερη), και txr (0.73% υψηλότερη). Ο Symmetric KNN παρουσιάζει τη χειρότερη απόδοση στα περισσότερα datasets, με τις μεγαλύτερες αποκλίσεις να εμφανίζονται στο lir (2.57% χαμηλότερη ακρίβεια) και το sh (21.9% χαμηλότερο F1-Score λόγω σημαντικής πτώσης στην ανάκληση). Αξιοσημείωτη είναι η περίπτωση του bl όπου ο Symmetric KNN υπερέχει με 2.4% υψηλότερη ακρίβεια. Ο Conventional KNN παραμένει ο ταχύτερος, με τον Mutual KNN και τον Symmetric KNN να απαιτούν περίπου 5-6 φορές περισσότερο χρόνο εκτέλεσης στα μεγαλύτερα datasets.

Πίνακας 6.4: Σύγκριση μεθόδων KNN με k=9

Dataset	Μετρική	Conventional KNN k=9	Mutual k=9	Symmetric k=9
	CPU time (sec)	5.5334	36.4896	37.0191
	Accuracy	0.836751	0.826236	0.817298
mgt	Avg Precision	0.839946	0.817950	0.830138
	Avg Recall	0.797051	0.794867	0.765205
	F1-Score	0.811217	0.803774	0.782202
	CPU time (sec)	94.8479	594.6001	291.9926
	Accuracy	0.998707	0.999483	0.997241
sh	Avg Precision	0.856589	0.908893	0.712331
	Avg Recall	0.781234	0.923980	0.519314
	F1-Score	0.813073	0.892375	0.559578
	CPU time (sec)	0.6738	6.5525	6.7550
	Accuracy	0.976351	0.963514	0.977027
tn	Avg Precision	0.976369	0.963539	0.977032
	Avg Recall	0.976356	0.963519	0.977025
	F1-Score	0.976351	0.963513	0.977027
	CPU time (sec)	0.0010	0.0021	0.0026
	Accuracy	0.933333	0.933333	0.933333
iris	Avg Precision	0.944444	0.944444	0.944444
	Avg Recall	0.933333	0.933333	0.933333
	F1-Score	0.932660	0.932660	0.932660
	CPU time (sec)	1.9590	14.8425	13.8168
	Accuracy	0.912199	0.915307	0.882673
ls	Avg Precision	0.906865	0.904074	0.880382
	Avg Recall	0.893348	0.899035	0.854786
	F1-Score	0.898944	0.900647	0.864378
	CPU time (sec)	0.1236	1.2633	1.5859

bn

Dataset	Μετρική	Conventional KNN k=9	Mutual k=9	Symmetric k=9
	Accuracy	0.901887	0.904717	0.904717
	Avg Precision	0.902503	0.905942	0.905942
	Avg Recall	0.899591	0.902022	0.902022
	F1-Score	0.900779	0.903542	0.903542
	CPU time (sec)	0.0029	0.0235	0.0224
	Accuracy	0.880000	0.880000	0.896000
bl	Avg Precision	0.590797	0.590797	0.597136
	Avg Recall	0.635735	0.635735	0.648555
	F1-Score	0.611614	0.611614	0.621773
	CPU time (sec)	0.0125	0.1350	0.1486
	Accuracy	0.597973	0.564189	0.631757
ys	Avg Precision	0.618667	0.597434	0.507851
	Avg Recall	0.565165	0.527072	0.459548
	F1-Score	0.584628	0.550326	0.464097
	CPU time (sec)	0.1885	1.7279	1.7073
	Accuracy	0.856481	0.876852	0.834259
ph	Avg Precision	0.832805	0.855791	0.806345
	Avg Recall	0.816994	0.845868	0.786801
	F1-Score	0.824165	0.850560	0.795333
	CPU time (sec)	1.7104	12.4384	10.8248
	Accuracy	0.987261	0.992266	0.979982
pd	Avg Precision	0.987667	0.992187	0.980625
	Avg Recall	0.987695	0.992251	0.981178
	F1-Score	0.987608	0.992210	0.980649
	CPU time (sec)	0.0008	0.0058	0.0069
	Accuracy	0.895522	0.880597	0.791045
ecl	Avg Precision	0.899348	0.762769	0.535473
	Avg Recall	0.911690	0.770040	0.485069
	F1-Score	0.903977	0.765076	0.486068
	CPU time (sec)	0.6915	5.6961	5.6027
	Accuracy	0.973636	0.990000	0.970000
txr	Avg Precision	0.974294	0.990297	0.970942
	Avg Recall	0.974778	0.990082	0.971321
	F1-Score	0.974326	0.990163	0.970711
	CPU time (sec)	0.0029	0.0269	0.0289
	Accuracy	0.803922	0.803922	0.771242
pm	Avg Precision	0.792812	0.785616	0.774923
	Avg Recall	0.761321	0.774623	0.700849
	F1-Score	0.772321	0.779412	0.714932
	CPU time (sec)	20.0736	79.3694	183.6676
	Accuracy	0.949000	0.957750	0.906750
lir	Avg Precision	0.948510	0.957140	0.907920
	Avg Recall	0.948558	0.957433	0.905845
	F1-Score	0.948138	0.957082	0.905834

Τα αποτελέσματα για $k = 9$ παρουσιάζονται στον Πίνακα 6.4. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε πέντε datasets (sh, ls, ph, pd, lir), με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο pd (0.50% υψηλότερη), ph (2.04% υψηλότερη), και txr (1.64% υψηλότερη). Ο Conventional KNN υπερτερεί στα mgt, ecl και pm, ενώ ο Symmetric KNN παρουσιάζει την καλύτερη απόδοση στα ys (3.38% υψηλότερη ακρίβεια), bl (1.6% υψηλότερη), bn (ισοπαλία με Mutual), και tn (0.07% υψηλότερη). Ο Symmetric KNN εμφανίζει σημαντική υποβάθμιση στο sh (25.3% χαμηλότερο F1-Score) και στο ecl (10.45% χαμηλότερη ακρίβεια). Ο Conventional KNN παραμένει ο ταχύτερος, με τον Mutual KNN να απαιτεί 6-8 φορές περισσότερο χρόνο, ενώ ο Symmetric KNN παρουσιάζει ακόμη μεγαλύτερη υπολογιστική επιβάρυνση στο lir (9 φορές αργότερος από τον Conventional).

Πίνακας 6.5: Σύγκριση μεθόδων KNN με βέλτιστο k ανά dataset - Δεδομένα χωρίς θόρυβο

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
bl	Best k	20	14	14
	CPU time (sec)	0.0037	0.0246	0.0277
	Accuracy	0.9040	0.8880	0.8880
	Average Precision	0.6031	0.5922	0.5914
	Average Recall	0.6550	0.6421	0.6433
	F1-Score	0.6279	0.6161	0.6162
ph	Best k	1	1	1
	CPU time (sec)	0.2164	0.4917	0.4182
	Accuracy	0.8870	0.8870	0.8704
	Average Precision	0.8728	0.8728	0.8516
	Average Recall	0.8513	0.8513	0.8314
	F1-Score	0.8609	0.8609	0.8404
txr	Best k	4	4	1
	CPU time (sec)	0.5748	2.7806	1.2702
	Accuracy	0.9873	0.9909	0.9864
	Average Precision	0.9874	0.9913	0.9866
	Average Recall	0.9878	0.9911	0.9869
	F1-Score	0.9875	0.9911	0.9867
iris	Best k	3	11	7
	CPU time (sec)	0.0003	0.0022	0.0017
	Accuracy	0.9333	0.9333	0.9333
	Average Precision	0.9444	0.9444	0.9444
	Average Recall	0.9333	0.9333	0.9333
	F1-Score	0.9327	0.9327	0.9327
bn	Best k	19	19	18
	CPU time (sec)	0.1399	2.9300	2.7824
	Accuracy	0.9066	0.9000	0.9028
	Average Precision	0.9084	0.9015	0.9057
	Average Recall	0.9036	0.8970	0.8991
	F1-Score	0.9054	0.8987	0.9014
pm	Best k	20	19	7
	CPU time (sec)	0.0039	0.0649	0.0408
	Accuracy	0.7843	0.7974	0.7647
	Average Precision	0.7821	0.7833	0.7621
	Average Recall	0.7242	0.7563	0.6958
	F1-Score	0.7386	0.7660	0.7089
ecl	Best k	3	8	3
	CPU time (sec)	0.0015	0.0064	0.0053
	Accuracy	0.8657	0.8806	0.9104
	Average Precision	0.7641	0.7628	0.9298
	Average Recall	0.7343	0.7700	0.8775
	F1-Score	0.7442	0.7651	0.8983
ys	Best k	12	18	5
	CPU time (sec)	0.0152	0.2619	0.0899
	Accuracy	0.6081	0.5946	0.5507
	Average Precision	0.6190	0.5976	0.4765
	Average Recall	0.5611	0.5709	0.4481
	F1-Score	0.5821	0.5797	0.4563
tn	Best k	18	20	20

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.5 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	0.6269	13.1124	13.1123
	Accuracy	0.9791	0.9743	0.9784
	Average Precision	0.9791	0.9743	0.9784
	Average Recall	0.9791	0.9743	0.9784
	F1-Score	0.9791	0.9743	0.9784
ls	Best k	16	16	3
	CPU time (sec)	0.7974	57.9781	2.5657
	Accuracy	0.9169	0.9169	0.9044
	Average Precision	0.9045	0.9085	0.8945
	Average Recall	0.8964	0.8964	0.8858
	F1-Score	0.9004	0.9001	0.8895
pd	Best k	8	8	1
	CPU time (sec)	2.4593	11.9027	2.7142
	Accuracy	0.9904	0.9927	0.9882
	Average Precision	0.9906	0.9926	0.9884
	Average Recall	0.9906	0.9928	0.9883
	F1-Score	0.9906	0.9927	0.9883
mgt	Best k	18	18	3
	CPU time (sec)	4.5226	57.4014	14.9898
	Accuracy	0.8370	0.8423	0.8289
	Average Precision	0.8428	0.8400	0.8374
	Average Recall	0.7954	0.8090	0.7830
	F1-Score	0.8105	0.8204	0.7991
lir	Best k	6	6	1
	CPU time (sec)	17.7065	99.5486	35.0651
	Accuracy	0.9580	0.9583	0.9418
	Average Precision	0.9574	0.9574	0.9409
	Average Recall	0.9575	0.9578	0.9412
	F1-Score	0.9572	0.9574	0.9408
sh	Best k	1	1	1
	CPU time (sec)	83.0354	166.2640	113.1003
	Accuracy	0.9996	0.9996	0.9992
	Average Precision	0.9131	0.9131	0.9173
	Average Recall	0.9241	0.9241	0.9107
	F1-Score	0.8945	0.8945	0.8894

Τα αποτελέσματα με βέλτιστο k για κάθε dataset παρουσιάζονται στον Πίνακα 6.5. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε επτά datasets (pm, ecl, txr, mgt, pd, lir, bn), με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο ecl (1.49% υψηλότερη ακρίβεια και 15.41% υψηλότερο F1-Score σε σχέση με τον Conventional KNN) και στο pm (1.31% υψηλότερη ακρίβεια). Ο Conventional KNN υπερτερεί στα bl (1.6% υψηλότερη), ys (1.35% υψηλότερη), και tn (0.48% υψηλότερη), ενώ στα iris, ls και sh οι δύο μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Ο Symmetric KNN παρουσιάζει γενικά χαμηλότερη απόδοση, με σημαντικές αποκλίσεις στο lir (1.62% χαμηλότερη), ys (5.74% χαμηλότερη), και pm (3.27% χαμηλότερη), αν και υπερέχει στο ecl (4.47% υψηλότερη από τον Conventional). Παρατηρείται ότι οι βέλτιστες τιμές του k διαφέρουν σημαντικά μεταξύ των μεθόδων και των datasets, με τον Symmetric KNN να προτιμά συχνότερα χαμηλότερες τιμές (π.χ. $k = 1$ στα txr, pd, lir, sh). Από πλευράς υπολογιστικής απόδοσης, ο Conventional KNN παραμένει ο ταχύτερος, ενώ ο Mutual KNN εμφανίζει σημαντική υπολογιστική επιβάρυνση στα μεγάλα datasets όπως το lir (5.6 φορές αργότερος) και το ls (72.7 φορές αργότερος).

Πίνακας 6.6: Σύγκριση μεθόδων KNN με adaptive $k_{\max} = 9$

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	5.3794	28.7874	17.5528
	Accuracy	0.830967	0.822818	0.823607
mgt				

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Average Precision	0.827314	0.814616	0.821527
	Average Recall	0.795717	0.790226	0.784533
	F1-Score	0.807062	0.799470	0.797008
	CPU time (sec)	237.7697	378.4281	341.7639
	Accuracy	0.999052	0.999655	0.999138
sh	Average Precision	0.917175	0.917551	0.917254
	Average Recall	0.901659	0.924076	0.901675
	F1-Score	0.884002	0.896749	0.884050
	CPU time (sec)	0.8458	3.2195	4.5741
	Accuracy	0.965541	0.961486	0.964189
tn	Average Precision	0.965550	0.961486	0.964191
	Average Recall	0.965538	0.961487	0.964188
	F1-Score	0.965540	0.961486	0.964189
	CPU time (sec)	0.0072	0.0057	0.0013
	Accuracy	0.933333	0.933333	0.933333
iris	Average Precision	0.944444	0.944444	0.944444
	Average Recall	0.933333	0.933333	0.933333
	F1-Score	0.932660	0.932660	0.932660
	CPU time (sec)	9.0652	22.2941	16.2066
	Accuracy	0.912199	0.912199	0.910645
ls	Average Precision	0.902396	0.900361	0.899246
	Average Recall	0.896054	0.898323	0.896330
	F1-Score	0.898517	0.898973	0.897448
	CPU time (sec)	0.2047	0.6488	0.7582
	Accuracy	0.896226	0.899057	0.892453
bn	Average Precision	0.896374	0.899734	0.892074
	Average Recall	0.894223	0.896654	0.890926
	F1-Score	0.895138	0.897896	0.891448
	CPU time (sec)	0.0131	0.0102	0.0225
	Accuracy	0.840000	0.840000	0.824000
bl	Average Precision	0.575808	0.575808	0.581864
	Average Recall	0.605922	0.605922	0.593101
	F1-Score	0.589770	0.589770	0.584488
	CPU time (sec)	0.0249	0.1366	0.1665
	Accuracy	0.584459	0.547297	0.564189
ys	Average Precision	0.619137	0.590646	0.576759
	Average Recall	0.574611	0.518501	0.549243
	F1-Score	0.588249	0.539350	0.555407
	CPU time (sec)	0.2592	0.9271	0.8254
	Accuracy	0.887963	0.892593	0.886111
ph	Average Precision	0.877180	0.881283	0.874780
	Average Recall	0.848380	0.856168	0.846164
	F1-Score	0.860812	0.867234	0.858511
	CPU time (sec)	2.4182	4.7282	3.7399
	Accuracy	0.988171	0.990901	0.989536
pd	Average Precision	0.988392	0.991009	0.989698
	Average Recall	0.988292	0.990991	0.989790
	F1-Score	0.988323	0.990987	0.989721
	CPU time (sec)	0.0016	0.0077	0.0029
	Accuracy	0.940299	0.865672	0.910448
ecl	Average Precision	0.949169	0.743290	0.912121
	Average Recall	0.929398	0.729861	0.880440

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	F1-Score	0.934921	0.731875	0.892720
	CPU time (sec)	7.2542	15.5175	12.0425
	Accuracy	0.986364	0.988182	0.986364
txr	Average Precision	0.986604	0.988248	0.986604
	Average Recall	0.987024	0.988685	0.987024
	F1-Score	0.986764	0.988414	0.986764
	CPU time (sec)	0.0089	0.0402	0.0409
	Accuracy	0.758170	0.764706	0.758170
pm	Average Precision	0.733333	0.740196	0.737130
	Average Recall	0.721887	0.735755	0.708585
	F1-Score	0.726588	0.737814	0.717585
	CPU time (sec)	43.6830	71.5831	73.7346
	Accuracy	0.954250	0.955500	0.947500
lir	Average Precision	0.953590	0.954750	0.946679
	Average Recall	0.953964	0.955080	0.946823
	F1-Score	0.953510	0.954751	0.946474

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = 9$ παρουσιάζονται στον Πίνακα 6.6. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε έξι datasets (sh, bn, ph, pd, txr, lir), με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο pd (0.27% υψηλότερη ακρίβεια), ph (0.46% υψηλότερη), και bn (0.28% υψηλότερη). Ο Conventional KNN υπερτερεί στα mgt (0.81% υψηλότερη), tn (0.41% υψηλότερη), ys (3.72% υψηλότερη), και ecl (7.46% υψηλότερη ακρίβεια και 20.3% υψηλότερο F1-Score), ενώ στα iris, ls και bl οι δύο μέθοδοι παρουσιάζουν την ίδια ακρίβεια. Ο Symmetric KNN εμφανίζει χαμηλότερη απόδοση στα περισσότερα datasets, με σημαντικές αποκλίσεις στο ecl (2.98% χαμηλότερη από τον Conventional), ys (2.03% χαμηλότερη), και lir (0.68% χαμηλότερη). Η υπολογιστική επιβάρυνση του Mutual KNN είναι μεγαλύτερη στα μεγάλα datasets, με σημαντικά αυξημένους χρόνους εκτέλεσης στο mgt (5.4 φορές αργότερος) και το sh (1.6 φορές αργότερος).

Πίνακας 6.7: Σύγκριση μεθόδων KNN με adaptive $k_{\max} = 20$

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	5.9577	21.2659	17.1700
	Accuracy	0.828601	0.829653	0.822555
mgt	Average Precision	0.825846	0.824413	0.820230
	Average Recall	0.791725	0.795693	0.783385
	F1-Score	0.803652	0.806253	0.795798
	CPU time (sec)	233.4486	352.8469	316.7628
	Accuracy	0.999052	0.999655	0.999138
sh	Average Precision	0.917175	0.917551	0.917254
	Average Recall	0.901659	0.924076	0.901675
	F1-Score	0.884002	0.896749	0.884050
	CPU time (sec)	1.3258	3.0746	2.2976
	Accuracy	0.965541	0.962162	0.964189
tn	Average Precision	0.965542	0.962162	0.964191
	Average Recall	0.965540	0.962162	0.964188
	F1-Score	0.965540	0.962162	0.964189
	CPU time (sec)	0.0061	0.0013	0.0060
	Accuracy	0.933333	0.966667	0.933333
iris	Average Precision	0.944444	0.969697	0.944444
	Average Recall	0.933333	0.966667	0.933333
	F1-Score	0.932660	0.966583	0.932660
	CPU time (sec)	11.5752	24.7349	3.4492
	Accuracy	0.910645	0.917638	0.909091
ls	Average Precision	0.900099	0.908466	0.897616
	Average Recall	0.894483	0.901762	0.894714

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	F1-Score	0.896647	0.904239	0.895839
	CPU time (sec)	0.1945	0.8785	0.7877
	Accuracy	0.897170	0.900000	0.892453
bn	Average Precision	0.897241	0.900387	0.892221
	Average Recall	0.895259	0.897858	0.890757
	F1-Score	0.896111	0.898911	0.891407
	CPU time (sec)	0.0078	0.0240	0.0218
	Accuracy	0.840000	0.840000	0.824000
bl	Average Precision	0.575808	0.575808	0.581864
	Average Recall	0.605922	0.605922	0.593101
	F1-Score	0.589770	0.589770	0.584488
	CPU time (sec)	0.0399	0.1638	0.1449
	Accuracy	0.570946	0.557432	0.547297
ys	Average Precision	0.582369	0.583955	0.542775
	Average Recall	0.553371	0.554454	0.483642
	F1-Score	0.560091	0.564091	0.496226
	CPU time (sec)	0.3915	0.9458	0.8842
	Accuracy	0.894444	0.890741	0.886111
ph	Average Precision	0.885978	0.879643	0.877109
	Average Recall	0.855688	0.853053	0.843467
	F1-Score	0.868730	0.864674	0.857638
	CPU time (sec)	2.4940	5.1586	3.7174
	Accuracy	0.988626	0.990901	0.989536
pd	Average Precision	0.988832	0.991064	0.989698
	Average Recall	0.988873	0.991082	0.989790
	F1-Score	0.988830	0.991056	0.989721
	CPU time (sec)	0.0019	0.0150	0.0049
	Accuracy	0.940299	0.880597	0.880597
ecl	Average Precision	0.952862	0.872222	0.885757
	Average Recall	0.916088	0.870023	0.828588
	F1-Score	0.931461	0.864172	0.852664
	CPU time (sec)	8.0113	2.4936	2.0319
	Accuracy	0.986364	0.988182	0.986364
txr	Average Precision	0.986604	0.988248	0.986604
	Average Recall	0.987024	0.988685	0.987024
	F1-Score	0.986764	0.988414	0.986764
	CPU time (sec)	0.0139	0.0738	0.0472
	Accuracy	0.771242	0.764706	0.758170
pm	Average Precision	0.749695	0.740581	0.737130
	Average Recall	0.731887	0.731321	0.708585
	F1-Score	0.738691	0.735294	0.717585
	CPU time (sec)	43.8698	90.7848	77.7929
	Accuracy	0.952500	0.955000	0.947250
lir	Average Precision	0.951862	0.954195	0.946558
	Average Recall	0.952007	0.954535	0.946588
	F1-Score	0.951630	0.954122	0.946274

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = 20$ παρουσιάζονται στον Πίνακα 6.7. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε επτά datasets (mgt, sh, iris, ls, bn, pd, lir, txr), με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο iris (3.33% υψηλότερη ακρίβεια), ls (0.69% υψηλότερη), και lir (0.25% υψηλότερη). Ο Conventional KNN υπερτερεί στα tn (0.34% υψηλότερη), ys (1.35% υψηλότερη), ph (0.37% υψηλότερη), pm (0.65% υψηλότερη), και ecl (5.97% υψηλότερη ακρίβεια και 6.73% υψηλότερο F1-Score), ενώ στα bl οι δύο μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Ο Symmetric KNN εμφανίζει χαμηλότερη απόδοση στα περισσότερα datasets, με

σημαντικές αποκλίσεις στο ecl (5.97% χαμηλότερη από τον Conventional), ys (2.36% χαμηλότερη), και lir (0.53% χαμηλότερη). Αξιοσημείωτη είναι η βελτίωση του Mutual KNN στα μικρότερα datasets όπως το iris και η σταθερή του υπεροχή στα pd και txr. Ο Conventional KNN παραμένει ο ταχύτερος στα περισσότερα datasets, με εξαίρεση το txr όπου ο Mutual KNN είναι 3.2 φορές ταχύτερος.

Πίνακας 6.8: Σύγκριση μεθόδων KNN με adaptive $k_{\max} = 50$

Dataset	**Μετρική**	**Conventional KNN**	**Mutual KNN**	**Symmetric KNN**
mgt	CPU time (sec)	5.6883	29.7655	25.8128
	Accuracy	0.829127	0.829653	0.822555
	Average Precision	0.826487	0.825176	0.820230
	Average Recall	0.792299	0.794864	0.783385
	F1-Score	0.804254	0.805848	0.795798
sh	CPU time (sec)	216.3003	292.6843	242.1617
	Accuracy	0.999052	0.999224	0.999138
	Average Precision	0.917175	0.917473	0.917254
	Average Recall	0.901659	0.901754	0.901675
	F1-Score	0.884002	0.884199	0.884050
tn	CPU time (sec)	1.5374	3.4691	2.2732
	Accuracy	0.966216	0.963514	0.964189
	Average Precision	0.966221	0.963518	0.964191
	Average Recall	0.966214	0.963512	0.964188
	F1-Score	0.966216	0.963513	0.964189
iris	CPU time (sec)	0.0007	0.0067	0.0024
	Accuracy	0.933333	0.933333	0.933333
	Average Precision	0.944444	0.944444	0.944444
	Average Recall	0.933333	0.933333	0.933333
	F1-Score	0.932660	0.932660	0.932660
ls	CPU time (sec)	1.9051	7.5913	4.6624
	Accuracy	0.911422	0.916861	0.909091
	Average Precision	0.901230	0.907924	0.897616
	Average Recall	0.895080	0.900290	0.894714
	F1-Score	0.897432	0.903097	0.895839
bn	CPU time (sec)	0.2562	1.1376	0.9950
	Accuracy	0.898113	0.900943	0.891509
	Average Precision	0.898473	0.901650	0.891344
	Average Recall	0.895956	0.898556	0.889722
	F1-Score	0.897004	0.899805	0.890434
bl	CPU time (sec)	0.0032	0.0094	0.0305
	Accuracy	0.840000	0.840000	0.824000
	Average Precision	0.575808	0.575808	0.581864
	Average Recall	0.605922	0.605922	0.593101
	F1-Score	0.589770	0.589770	0.584488
ys	CPU time (sec)	0.0533	0.3897	0.2760
	Accuracy	0.570946	0.577703	0.554054
	Average Precision	0.577726	0.584566	0.544986
	Average Recall	0.542728	0.562383	0.483672
	F1-Score	0.550088	0.569254	0.496669
ph	CPU time (sec)	0.3967	1.3258	0.9874
	Accuracy	0.896296	0.887037	0.887037
	Average Precision	0.890060	0.876358	0.877922
	Average Recall	0.856107	0.846823	0.845025
	F1-Score	0.870503	0.859518	0.858941
pd	CPU time (sec)	2.0653	5.0136	4.4054
	Accuracy	0.987716	0.989991	0.989081

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Average Precision	0.988022	0.990259	0.989406
	Average Recall	0.987956	0.990267	0.989404
	F1-Score	0.987963	0.990242	0.989382
	CPU time (sec)	0.0069	0.0219	0.0206
	Accuracy	0.910448	0.865672	0.895522
ecl	Average Precision	0.937840	0.852083	0.907870
	Average Recall	0.872338	0.823380	0.833796
	F1-Score	0.900117	0.834414	0.865290
	CPU time (sec)	1.5067	2.6027	3.6600
	Accuracy	0.986364	0.987273	0.986364
txr	Average Precision	0.986604	0.987552	0.986604
	Average Recall	0.987024	0.987865	0.987024
	F1-Score	0.986764	0.987659	0.986764
	CPU time (sec)	0.0153	0.0706	0.0512
	Accuracy	0.764706	0.764706	0.758170
pm	Average Precision	0.741369	0.740196	0.737130
	Average Recall	0.726887	0.735755	0.708585
	F1-Score	0.732621	0.737814	0.717585
	CPU time (sec)	48.0132	102.8258	80.8520
	Accuracy	0.950750	0.952750	0.946500
lir	Average Precision	0.950049	0.951910	0.945898
	Average Recall	0.950261	0.952145	0.945798
	F1-Score	0.949852	0.951794	0.945516

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = 50$ παρουσιάζονται στον Πίνακα 6.8. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε εννέα datasets (mgt, sh, ls, bn, ys, pd, txr, lir, pm), με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο pd (0.23% υψηλότερη ακρίβεια), ys (0.68% υψηλότερη), ls (0.54% υψηλότερη), και lir (0.20% υψηλότερη). Ο Conventional KNN υπερτερεί στα tn (0.27% υψηλότερη), ph (0.93% υψηλότερη), και ecl (4.48% υψηλότερη ακρίβεια και 6.57% υψηλότερο F1-Score), ενώ στα iris και bl οι δύο μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Ο Symmetric KNN παρουσιάζει χαμηλότερη απόδοση στα περισσότερα datasets, με σημαντικές αποκλίσεις στο ecl (1.49% χαμηλότερη από τον Conventional), ys (1.69% χαμηλότερη), και lir (0.43% χαμηλότερη). Αξιοσημείωτη είναι η σταθερή υπεροχή του Mutual KNN στα περισσότερα datasets με την αύξηση του $k_{\max}$ στο 50, υποδεικνύοντας την αποτελεσματικότητα της μεθόδου σε μεγαλύτερα εύρη γειτόνων. Ο Conventional KNN παραμένει ο ταχύτερος στα περισσότερα datasets, με τον Mutual KNN να απαιτεί 2-5 φορές περισσότερο χρόνο στα μεγάλα datasets.

Πίνακας 6.9: Σύγκριση μεθόδων KNN με adaptive $k_{\max} = \sqrt{n/2}$

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	7.9342	47.1603	40.2073
	Accuracy	0.826761	0.829653	0.822029
mgt	Average Precision	0.824022	0.825176	0.819756
	Average Recall	0.789303	0.794864	0.782646
	F1-Score	0.801330	0.805848	0.795103
	CPU time (sec)	64.9979	96.6594	78.8166
	Accuracy	0.999052	0.999224	0.999138
sh	Average Precision	0.917175	0.917473	0.917254
	Average Recall	0.901659	0.901754	0.901675
	F1-Score	0.884002	0.884199	0.884050
	CPU time (sec)	1.4451	5.6282	6.6120
	Accuracy	0.966216	0.963514	0.964189
tn	Average Precision	0.966221	0.963518	0.964191
	Average Recall	0.966214	0.963512	0.964188
	F1-Score	0.966216	0.963513	0.964189

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	0.0016	0.0012	0.0021
	Accuracy	0.900000	0.933333	0.933333
iris	Average Precision	0.902357	0.944444	0.944444
	Average Recall	0.900000	0.933333	0.933333
	F1-Score	0.899749	0.932660	0.932660
	CPU time (sec)	2.3612	14.5800	8.6463
	Accuracy	0.911422	0.916861	0.909091
ls	Average Precision	0.901230	0.908344	0.897616
	Average Recall	0.895080	0.899593	0.894714
	F1-Score	0.897432	0.902766	0.895839
	CPU time (sec)	0.1881	1.1472	0.9358
	Accuracy	0.897170	0.900943	0.891509
bn	Average Precision	0.897421	0.901650	0.891344
	Average Recall	0.895090	0.898556	0.889722
	F1-Score	0.896071	0.899805	0.890434
	CPU time (sec)	0.0114	0.0093	0.0148
	Accuracy	0.840000	0.840000	0.824000
bl	Average Precision	0.575808	0.575808	0.581864
	Average Recall	0.605922	0.605922	0.593101
	F1-Score	0.589770	0.589770	0.584488
	CPU time (sec)	0.0192	0.2671	0.1796
	Accuracy	0.570946	0.560811	0.550676
ys	Average Precision	0.573076	0.596984	0.546942
	Average Recall	0.544079	0.555604	0.500309
	F1-Score	0.550437	0.567564	0.504797
	CPU time (sec)	0.2302	1.5609	2.8739
	Accuracy	0.894444	0.886111	0.886111
ph	Average Precision	0.886795	0.874780	0.876312
	Average Recall	0.854789	0.846164	0.844366
	F1-Score	0.868462	0.858511	0.857931
	CPU time (sec)	3.6930	10.7483	7.1167
	Accuracy	0.987261	0.989991	0.989081
pd	Average Precision	0.987597	0.990259	0.989406
	Average Recall	0.987570	0.990267	0.989404
	F1-Score	0.987556	0.990242	0.989382
	CPU time (sec)	0.0026	0.0039	0.0083
	Accuracy	0.925373	0.865672	0.895522
ecl	Average Precision	0.943058	0.851389	0.894066
	Average Recall	0.882755	0.836690	0.847106
	F1-Score	0.907892	0.838255	0.865793
	CPU time (sec)	1.5901	4.8540	3.6227
	Accuracy	0.986364	0.988182	0.986364
txr	Average Precision	0.986604	0.988372	0.986604
	Average Recall	0.987024	0.988715	0.987024
	F1-Score	0.986764	0.988490	0.986764
	CPU time (sec)	0.0062	0.0283	0.0453
	Accuracy	0.771242	0.771242	0.758170
pm	Average Precision	0.749695	0.747767	0.737130
	Average Recall	0.731887	0.740755	0.708585
	F1-Score	0.738691	0.743890	0.717585
	CPU time (sec)	16.8195	39.8150	33.0699
	Accuracy	0.949750	0.951500	0.946500
lir	Average Precision	0.949177	0.950696	0.945913

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Average Recall	0.949248	0.950890	0.945758
	F1-Score	0.948886	0.950511	0.945465

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = \sqrt{n/2}$ παρουσιάζονται στον Πίνακα 6.9. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε οκτώ datasets (mgt, sh, iris, ls, bn, pd, txr, lir), με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο iris (3.33% υψηλότερη ακρίβεια), pd (0.27% υψηλότερη), ls (0.54% υψηλότερη), και mgt (0.29% υψηλότερη). Ο Conventional KNN υπερτερεί στα tn (0.27% υψηλότερη), ys (1.01% υψηλότερη), ph (0.83% υψηλότερη), και ecl (5.97% υψηλότερη ακρίβεια και 6.96% υψηλότερο F1-Score), ενώ στα pm και bl οι δύο μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Ο Symmetric KNN εμφανίζει χαμηλότερη απόδοση στα περισσότερα datasets, με σημαντικές αποκλίσεις στο ecl (2.99% χαμηλότερη από τον Conventional), ys (2.03% χαμηλότερη), και ph (0.83% χαμηλότερη). Η χρήση του $k_{\max} = \sqrt{n/2}$ προσαρμόζει δυναμικά το εύρος των γειτόνων ανάλογα με το μέγεθος του dataset, με τον Mutual KNN να επωφελείται ιδιαίτερα σε datasets μεσαίου μεγέθους. Ο Conventional KNN παραμένει ο ταχύτερος, με τον Mutual KNN να απαιτεί 2-6 φορές περισσότερο χρόνο στα μεγάλα datasets όπως το mgt και το lir.

Πίνακας 6.10: Σύγκριση μεθόδων KNN με adaptive $k_{\max} = 20$, lastk window=3

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	7.6650	123.5421	118.4846
	Accuracy	0.832019	0.838591	0.812040
mgt	Average Precision	0.837673	0.834638	0.823534
	Average Recall	0.789067	0.805945	0.759136
	F1-Score	0.804239	0.816647	0.775709
	CPU time (sec)	101.5080	655.4334	607.5142
	Accuracy	0.998793	0.999397	0.998189
sh	Average Precision	0.927910	0.904946	0.784002
	Average Recall	0.857111	0.923965	0.755841
	F1-Score	0.863599	0.890303	0.747071
	CPU time (sec)	1.0829	33.4100	35.1786
	Accuracy	0.979730	0.975000	0.977703
tn	Average Precision	0.979732	0.975000	0.977703
	Average Recall	0.979732	0.975001	0.977704
	F1-Score	0.979730	0.975000	0.977703
	CPU time (sec)	0.0022	0.0071	0.0061
	Accuracy	0.933333	0.966667	0.933333
iris	Average Precision	0.944444	0.969697	0.944444
	Average Recall	0.933333	0.966667	0.933333
	F1-Score	0.932660	0.966583	0.932660
	CPU time (sec)	1.6770	43.4340	40.0946
	Accuracy	0.912199	0.926185	0.877234
ls	Average Precision	0.908052	0.921393	0.874730
	Average Recall	0.892599	0.907744	0.849307
	F1-Score	0.899046	0.912134	0.858935
	CPU time (sec)	0.1825	4.6009	5.0279
	Accuracy	0.900943	0.900943	0.904717
bn	Average Precision	0.901443	0.901443	0.906190
	Average Recall	0.898725	0.898725	0.901854
	F1-Score	0.899845	0.899845	0.903501
	CPU time (sec)	0.0116	0.0453	0.1178
	Accuracy	0.904000	0.896000	0.896000
bl	Average Precision	0.603088	0.598268	0.598268
	Average Recall	0.654965	0.648555	0.648555
	F1-Score	0.627946	0.622303	0.622303
	CPU time (sec)	0.0150	0.3711	0.4823

ys

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Accuracy	0.601351	0.591216	0.614865
	Average Precision	0.582846	0.589915	0.574795
	Average Recall	0.552618	0.571719	0.494271
	F1-Score	0.561246	0.575571	0.506575
	CPU time (sec)	0.2293	5.9317	6.8274
	Accuracy	0.872222	0.873148	0.837963
ph	Average Precision	0.855183	0.852692	0.812390
	Average Recall	0.831788	0.838739	0.788538
	F1-Score	0.842054	0.845185	0.798694
	CPU time (sec)	3.3943	57.6468	45.0992
	Accuracy	0.984986	0.993631	0.974522
pd	Average Precision	0.985498	0.993562	0.975670
	Average Recall	0.985668	0.993720	0.975666
	F1-Score	0.985475	0.993619	0.975291
	CPU time (sec)	0.0016	0.0172	0.0342
	Accuracy	0.910448	0.895522	0.805970
ecl	Average Precision	0.929779	0.920346	0.706769
	Average Recall	0.877546	0.885648	0.549884
	F1-Score	0.898345	0.893506	0.591402
	CPU time (sec)	1.1535	31.8510	34.6165
	Accuracy	0.974545	0.990000	0.953636
txr	Average Precision	0.975262	0.990036	0.955111
	Average Recall	0.975862	0.990601	0.954812
	F1-Score	0.975288	0.990275	0.954369
	CPU time (sec)	0.0112	0.1390	0.1694
	Accuracy	0.810458	0.797386	0.758170
pm	Average Precision	0.807080	0.780780	0.777862
	Average Recall	0.761887	0.760755	0.673113
	F1-Score	0.776011	0.768555	0.684325
	CPU time (sec)	16.4720	154.2723	143.4812
	Accuracy	0.936500	0.954500	0.870500
lir	Average Precision	0.936410	0.953883	0.875329
	Average Recall	0.936031	0.953951	0.869195
	F1-Score	0.935570	0.953610	0.869961

Τα αποτελέσματα με τη μέθοδο adaptive KNN με $k_{\max} = 20$ και παράθυρο lastk window=3 παρουσιάζονται στον Πίνακα 6.10. Ο Mutual KNN επιτυγχάνει την υψηλότερη ακρίβεια σε οκτώ datasets (mgt, sh, iris, ls, ph, pd, txr, lir), με τις πιο εντυπωσιακές βελτιώσεις να εμφανίζονται στο pd (0.86% υψηλότερη ακρίβεια), txr (1.55% υψηλότερη), lir (1.80% υψηλότερη), ls (1.40% υψηλότερη), και mgt (0.66% υψηλότερη). Ο Conventional KNN υπερτερεί στα tn (0.47% υψηλότερη), bl (0.80% υψηλότερη), pm (1.31% υψηλότερη), και ecl (1.49% υψηλότερη), ενώ στα bn οι δύο μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Ο Symmetric KNN εμφανίζει την καλύτερη απόδοση μόνο στο ys (1.35% υψηλότερη από τον Conventional) και στο bn (0.38% υψηλότερη), αλλά παρουσιάζει σημαντική υποβάθμιση στα περισσότερα datasets, ιδιαίτερα στο ecl (10.45% χαμηλότερη ακρίβεια και 30.69% χαμηλότερο F1-Score) και στο lir (6.60% χαμηλότερη). Η χρήση του παραθύρου lastk window=3 φαίνεται να βελτιώνει σημαντικά την απόδοση του Mutual KNN σε datasets με πολύπλοκη δομή, αν και με σημαντική υπολογιστική επιβάρυνση—ο Mutual KNN απαιτεί 16-30 φορές περισσότερο χρόνο από τον Conventional KNN στα μεγάλα datasets (mgt, sh, lir, pd, txr).

Μετρήσεις στα 14 datasets με θόρυβο 30%

Πίνακας 6.11: Σύγκριση μεθόδων KNN με $k = 1$ και θόρυβο 30%

Dataset	Μετρική	Conventional KNN k=1	Mutual k=1	Symmetric k=1
	CPU time (sec)	0.0027	0.0045	0.0046
	Accuracy	0.616000	0.616000	0.600000
bl				

Dataset	Μετρική	Conventional KNN k=1	Mutual k=1	Symmetric k=1
	Average Precision	0.469979	0.469979	0.503655
	Average Recall	0.443223	0.443223	0.461803
	F1-Score	0.455782	0.455782	0.476552
	CPU time (sec)	0.1948	0.4353	0.3636
	Accuracy	0.692593	0.692593	0.680556
ph	Average Precision	0.648388	0.648388	0.642532
	Average Recall	0.664438	0.664438	0.662166
	F1-Score	0.652708	0.652708	0.645642
	CPU time (sec)	0.6592	1.3770	0.9697
	Accuracy	0.710000	0.710000	0.710000
txr	Average Precision	0.712428	0.712428	0.710631
	Average Recall	0.708663	0.708663	0.709880
	F1-Score	0.709602	0.709602	0.709339
	CPU time (sec)	0.0003	0.0005	0.0005
	Accuracy	0.633333	0.633333	0.666667
iris	Average Precision	0.636111	0.636111	0.708333
	Average Recall	0.633333	0.633333	0.666667
	F1-Score	0.630640	0.630640	0.675926
	CPU time (sec)	0.6700	1.5938	1.2628
	Accuracy	0.676351	0.676351	0.680405
tn	Average Precision	0.676354	0.676354	0.681124
	Average Recall	0.676354	0.676354	0.680362
	F1-Score	0.676351	0.676351	0.680055
	CPU time (sec)	0.0027	0.0108	0.0077
	Accuracy	0.607843	0.607843	0.640523
pm	Average Precision	0.597303	0.597303	0.625000
	Average Recall	0.606887	0.606887	0.636321
	F1-Score	0.593229	0.593229	0.624079
	CPU time (sec)	0.0007	0.0015	0.0014
	Accuracy	0.626866	0.626866	0.641791
ecl	Average Precision	0.415693	0.415693	0.610859
	Average Recall	0.404427	0.404427	0.498785
	F1-Score	0.405517	0.405517	0.536414
	CPU time (sec)	0.1114	0.2887	0.2333
	Accuracy	0.640566	0.640566	0.633962
bn	Average Precision	0.641086	0.641086	0.632677
	Average Recall	0.642186	0.642186	0.633591
	F1-Score	0.640027	0.640027	0.632623
	CPU time (sec)	0.0184	0.0311	0.0362
	Accuracy	0.378378	0.378378	0.375000
ys	Average Precision	0.270573	0.270573	0.267527
	Average Recall	0.398252	0.398252	0.376901
	F1-Score	0.280480	0.280480	0.274663
	CPU time (sec)	0.8263	2.2745	1.8690
	Accuracy	0.635587	0.635587	0.627040
ls	Average Precision	0.614535	0.614535	0.608908
	Average Recall	0.627088	0.627088	0.621688
	F1-Score	0.618632	0.618632	0.612543
	CPU time (sec)	2.0231	5.2259	3.4296
	Accuracy	0.692448	0.692448	0.702457
pd	Average Precision	0.691849	0.691849	0.702390
	Average Recall	0.693477	0.693477	0.702026

Dataset	Μετρική	Conventional KNN k=1	Mutual k=1	Symmetric k=1
	F1-Score	0.691778	0.691778	0.701361
	CPU time (sec)	4.2337	12.0866	9.6861
	Accuracy	0.618822	0.618822	0.612776
mgt	Average Precision	0.600944	0.600944	0.595101
	Average Recall	0.608097	0.608097	0.601914
	F1-Score	0.600719	0.600719	0.594622
	CPU time (sec)	16.6248	30.3312	38.2657
	Accuracy	0.686000	0.686000	0.658500
lir	Average Precision	0.686002	0.686002	0.658785
	Average Recall	0.685996	0.685996	0.657836
	F1-Score	0.685084	0.685084	0.656903
	CPU time (sec)	98.9884	214.1123	189.1575
	Accuracy	0.698250	0.698250	0.695319
sh	Average Precision	0.316054	0.316054	0.311968
	Average Recall	0.742906	0.742906	0.657801
	F1-Score	0.313918	0.313918	0.308884

Τα αποτελέσματα με θόρυβο 30% για $k = 1$ παρουσιάζονται στον Πίνακα 6.11. Παρατηρείται ότι ο Conventional KNN και ο Mutual KNN επιτυγχάνουν ταυτόσημη ακρίβεια σε όλα τα datasets, όπως και στα καθαρά δεδομένα. Ο Symmetric KNN υπερτερεί σε πέντε datasets: iris (3.33% υψηλότερη ακρίβεια), pm (3.27% υψηλότερη), ecl (1.49% υψηλότερη και 13.09% υψηλότερο F1-Score), pd (1.00% υψηλότερη), και tn (0.40% υψηλότερη). Αντίθετα, ο Symmetric KNN παρουσιάζει χαμηλότερη απόδοση στα υπόλοιπα datasets, με τις μεγαλύτερες αποκλίσεις να εμφανίζονται στο lir (2.75% χαμηλότερη ακρίβεια), bl (1.6% χαμηλότερη), και ph (1.20% χαμηλότερη). Στα txr οι τρεις μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Η παρουσία θορύβου μειώνει σημαντικά την ακρίβεια όλων των μεθόδων συγκριτικά με τα καθαρά δεδομένα, με τη μείωση να κυμαίνεται από 18

Πίνακας 6.12: Σύγκριση μεθόδων KNN με $k = 5$ και θόρυβο 30%

Dataset	Μετρική	Conventional KNN k=5	Mutual k=5	Symmetric k=5
	CPU time (sec)	0.0033	0.0146	0.0138
	Accuracy	0.776000	0.784000	0.872000
bl	Average Precision	0.599846	0.602361	0.582684
	Average Recall	0.588278	0.593569	0.631563
	F1-Score	0.593006	0.597219	0.606048
	CPU time (sec)	0.1782	1.1564	0.9128
	Accuracy	0.740741	0.722222	0.767593
ph	Average Precision	0.695881	0.680389	0.721454
	Average Recall	0.710379	0.699900	0.717798
	F1-Score	0.701403	0.686182	0.719550
	CPU time (sec)	0.4633	3.4131	3.1416
	Accuracy	0.935455	0.842727	0.975455
txr	Average Precision	0.935781	0.844199	0.975734
	Average Recall	0.935661	0.841741	0.976109
	F1-Score	0.935516	0.842538	0.975794
	CPU time (sec)	0.0002	0.0010	0.0016
	Accuracy	0.900000	0.766667	0.966667
iris	Average Precision	0.907407	0.783150	0.969697
	Average Recall	0.900000	0.766667	0.966667
	F1-Score	0.899522	0.762830	0.966583
	CPU time (sec)	0.6644	3.5257	3.6892
	Accuracy	0.793919	0.714865	0.937838
tn	Average Precision	0.793919	0.714879	0.938004
	Average Recall	0.793919	0.714870	0.937825
	F1-Score	0.793919	0.714863	0.937831
	CPU time (sec)	0.0028	0.0160	0.0214
pm				

Dataset	Μετρική	Conventional KNN k=5	Mutual k=5	Symmetric k=5
	Accuracy	0.653595	0.679739	0.686275
	Average Precision	0.638235	0.656239	0.655380
	Average Recall	0.650755	0.666321	0.658019
	F1-Score	0.637748	0.658684	0.656566
	CPU time (sec)	0.0007	0.0036	0.0039
	Accuracy	0.865672	0.791045	0.835821
ecl	Average Precision	0.781981	0.714657	0.706564
	Average Recall	0.765575	0.698810	0.669444
	F1-Score	0.771051	0.704762	0.684797
	CPU time (sec)	0.1207	0.7308	0.7116
	Accuracy	0.707547	0.690566	0.789623
bn	Average Precision	0.706458	0.688998	0.787995
	Average Recall	0.707927	0.690137	0.789220
	F1-Score	0.706543	0.689211	0.788448
	CPU time (sec)	0.0140	0.0635	0.0647
	Accuracy	0.459459	0.418919	0.537162
ys	Average Precision	0.393464	0.327400	0.471328
	Average Recall	0.475944	0.437128	0.425886
	F1-Score	0.414709	0.336513	0.444504
	CPU time (sec)	1.0235	6.4988	6.5155
	Accuracy	0.854701	0.736597	0.898990
ls	Average Precision	0.837596	0.711907	0.892527
	Average Recall	0.837711	0.718798	0.877851
	F1-Score	0.837328	0.714842	0.883900
	CPU time (sec)	1.3385	11.3480	11.3731
	Accuracy	0.935396	0.833940	0.983621
pd	Average Precision	0.934829	0.832920	0.984297
	Average Recall	0.936298	0.834398	0.984437
	F1-Score	0.935405	0.833183	0.984245
	CPU time (sec)	4.9618	29.3953	48.8518
	Accuracy	0.701104	0.656677	0.773659
mgt	Average Precision	0.678181	0.637837	0.755637
	Average Recall	0.685361	0.646917	0.738655
	F1-Score	0.680728	0.639078	0.745115
	CPU time (sec)	19.4947	59.7875	115.1148
	Accuracy	0.907250	0.824500	0.920000
lir	Average Precision	0.906652	0.824241	0.919369
	Average Recall	0.907000	0.823873	0.919072
	F1-Score	0.906360	0.823525	0.918684
	CPU time (sec)	102.0951	277.7251	472.5905
	Accuracy	0.937839	0.857660	0.987154
sh	Average Precision	0.442022	0.384301	0.551596
	Average Recall	0.855297	0.836191	0.657303
	F1-Score	0.473011	0.401843	0.586307

Τα αποτελέσματα με θόρυβο 30% για $k = 5$ παρουσιάζονται στον Πίνακα 6.12. Σε αντίθεση με τα καθαρά δεδομένα, ο Symmetric KNN επιτυγχάνει εντυπωσιακή υπεροχή σε εννέα από τα δεκατέσσερα datasets, με τις πιο σημαντικές βελτιώσεις να εμφανίζονται στο pd (4.82% υψηλότερη ακρίβεια), sh (4.93% υψηλότερη), tn (14.39% υψηλότερη), ls (4.43% υψηλότερη), bn (8.21% υψηλότερη), bl (9.6% υψηλότερη), iris (6.67% υψηλότερη), ys (7.77% υψηλότερη), και txr (4.00% υψηλότερη). Ο Conventional KNN υπερτερεί μόνο στο ecl (2.99% υψηλότερη ακρίβεια), ενώ ο Mutual KNN παρουσιάζει τη χειρότερη απόδοση σε όλα τα datasets εκτός από το pm όπου επιτυγχάνει 2.61% υψηλότερη ακρίβεια από τον Conventional. Αξιοσημείωτη είναι η ικανότητα του Symmetric KNN να αντιμετωπίζει αποτελεσματικά τον θόρυβο μέσω του συμμετρικού φιλτραρίσματος των γειτόνων, με αποτέλεσμα σημαντικές

βελτιώσεις σε datasets όπως το pd (4.8% απόλυτη βελτίωση) και το tn (14.4% απόλυτη βελτίωση). Ο Conventional KNN παραμένει ο ταχύτερος, αν και ο Symmetric KNN εμφανίζει σημαντικά αυξημένους χρόνους εκτέλεσης σε μεγάλα datasets όπως το sh και το lir.

Πίνακας 6.13: Σύγκριση μεθόδων KNN με $k = 9$ και θόρυβο 30%

Dataset	**Μετρική**	**Conventional KNN k=9**	**Mutual k=9**	**Symmetric k=9**
	CPU time (sec)	0.0035	0.0190	0.0222
	Accuracy	0.808000	0.808000	0.888000
bl	Average Precision	0.612499	0.612499	0.593553
	Average Recall	0.611681	0.611681	0.641026
	F1-Score	0.610673	0.610673	0.615873
	CPU time (sec)	0.0008	0.0061	0.0065
	Accuracy	0.820896	0.850746	0.791045
ecl	Average Precision	0.692593	0.747572	0.540727
	Average Recall	0.655903	0.737004	0.507986
	F1-Score	0.669993	0.739394	0.512302
	CPU time (sec)	0.7524	5.6298	5.7081
	Accuracy	0.967273	0.927273	0.967273
txr	Average Precision	0.967348	0.928406	0.968417
	Average Recall	0.968166	0.927385	0.968886
	F1-Score	0.967584	0.927490	0.968045
	CPU time (sec)	1.0526	10.3285	12.2557
	Accuracy	0.891220	0.811189	0.878011
ls	Average Precision	0.881887	0.788269	0.874323
	Average Recall	0.871016	0.798999	0.852316
	F1-Score	0.874980	0.792689	0.860601
	CPU time (sec)	0.0046	0.0252	0.0265
	Accuracy	0.692810	0.679739	0.725490
pm	Average Precision	0.675026	0.658883	0.695370
	Average Recall	0.689623	0.670755	0.679151
	F1-Score	0.676853	0.660969	0.684753
	CPU time (sec)	0.6336	8.9904	8.8100
	Accuracy	0.851351	0.747973	0.968919
tn	Average Precision	0.851353	0.748027	0.968936
	Average Recall	0.851353	0.747983	0.968915
	F1-Score	0.851351	0.747964	0.968918
	CPU time (sec)	0.0171	0.1065	0.1088
	Accuracy	0.547297	0.503378	0.584459
ys	Average Precision	0.554381	0.419368	0.448749
	Average Recall	0.490364	0.493514	0.408689
	F1-Score	0.512264	0.424468	0.424607
	CPU time (sec)	0.1746	1.4967	1.6672
	Accuracy	0.772222	0.756481	0.800926
ph	Average Precision	0.727384	0.713287	0.763164
	Average Recall	0.727384	0.728769	0.749603
	F1-Score	0.727384	0.719314	0.755624
	CPU time (sec)	0.0003	0.0017	0.0020
	Accuracy	0.933333	0.833333	0.933333
iris	Average Precision	0.936364	0.837121	0.944444
	Average Recall	0.933333	0.833333	0.933333
	F1-Score	0.933250	0.830688	0.932660
	CPU time (sec)	0.1232	1.1452	1.1573
	Accuracy	0.775472	0.755660	0.853774
bn	Average Precision	0.773710	0.754174	0.852388

Dataset	Μετρική	Conventional KNN k=9	Mutual k=9	Symmetric k=9
	Average Recall	0.774535	0.755663	0.853542
	F1-Score	0.774053	0.754559	0.852872
pd	CPU time (sec)	1.7194	15.5652	15.5965
	Accuracy	0.979982	0.920837	0.979072
	Average Precision	0.980190	0.920697	0.979874
	Average Recall	0.980332	0.922406	0.980231
	F1-Score	0.980179	0.921287	0.979806
lir	CPU time (sec)	19.8923	79.5499	79.8097
	Accuracy	0.934750	0.892500	0.899250
	Average Precision	0.933976	0.891338	0.900426
	Average Recall	0.934020	0.891835	0.898231
	F1-Score	0.933556	0.891215	0.897985
mgt	CPU time (sec)	4.9835	76.7905	45.7556
	Accuracy	0.737645	0.691640	0.805994
	Average Precision	0.714385	0.669902	0.803678
	Average Recall	0.717521	0.678352	0.761414
	F1-Score	0.715828	0.672439	0.774376
sh	CPU time (sec)	95.9908	591.8064	576.3621
	Accuracy	0.988447	0.936977	0.994482
	Average Precision	0.560492	0.440601	0.589348
	Average Recall	0.776997	0.873331	0.522342
	F1-Score	0.591579	0.472012	0.545980

Τα αποτελέσματα με θόρυβο 30% για $k = 9$ παρουσιάζονται στον Πίνακα 6.13. Ο Symmetric KNN συνεχίζει να επιδεικνύει εξαιρετική απόδοση παρουσία θορύβου, επιτυγχάνοντας την υψηλότερη ακρίβεια σε οκτώ datasets: tn (11.76% υψηλότερη από τον Conventional), bn (7.83% υψηλότερη), bl (8.00% υψηλότερη), mgt (6.83% υψηλότερη), ys (3.72% υψηλότερη), ph (2.87% υψηλότερη), pm (3.27% υψηλότερη), και sh (0.60% υψηλότερη). Ο Conventional KNN υπερτερεί σε τέσσερα datasets: pd (0.09% υψηλότερη), lir (3.55% υψηλότερη), ls (1.32% υψηλότερη), και iris (ισοπαλία με Symmetric), ενώ παρουσιάζει ίδια απόδοση με τον Symmetric στο txr. Ο Mutual KNN επιτυγχάνει την καλύτερη απόδοση μόνο στο ecl (2.98% υψηλότερη από τον Conventional και 5.97% υψηλότερη από τον Symmetric). Αξιοσημείωτη είναι η εξαιρετική βελτίωση του Symmetric KNN στο tn (11.76% απόλυτη βελτίωση), υποδεικνύοντας την ικανότητά του να φιλτράρει αποτελεσματικά τον θόρυβο σε υψηλοδιάστατα datasets με μέτριο k. Ο Conventional KNN παραμένει ο ταχύτερος, με τον Mutual και Symmetric KNN να απαιτούν σημαντικά περισσότερο χρόνο, ιδιαίτερα στα μεγάλα datasets όπως το mgt και το sh.

Πίνακας 6.14: Σύγκριση μεθόδων KNN με βέλτιστο k ανά dataset - Δεδομένα με θόρυβο 30%

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
iris	Best k	17	20	5
	CPU time (sec)	0.0005	0.0032	0.0012
	Accuracy	0.9667	0.9000	0.9667
	Average Precision	0.9697	0.8993	0.9697
	Average Recall	0.9667	0.9000	0.9667
	F1-Score	0.9658	0.8982	0.9658
bl	Best k	19	18	13
	CPU time (sec)	0.0032	0.0350	0.0297
	Accuracy	0.8800	0.8720	0.9040
	Average Precision	0.7578	0.5898	0.6016
	Average Recall	0.6671	0.6327	0.6550
	F1-Score	0.6656	0.6100	0.6270
bn	Best k	19	17	20
	CPU time (sec)	0.1281	2.4393	2.9124
	Accuracy	0.8321	0.7972	0.8925

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.14 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Average Precision	0.8306	0.7961	0.8924
	Average Recall	0.8321	0.7982	0.8906
	F1-Score	0.8312	0.7965	0.8914
ls	Best k	11	20	5
	CPU time (sec)	0.6869	16.6990	5.2560
	Accuracy	0.8998	0.8679	0.8990
	Average Precision	0.8925	0.8531	0.8925
	Average Recall	0.8782	0.8567	0.8779
	F1-Score	0.8836	0.8540	0.8839
ys	Best k	20	20	5
	CPU time (sec)	0.0195	0.2867	0.0773
	Accuracy	0.5878	0.5405	0.5372
	Average Precision	0.4849	0.5262	0.4713
	Average Recall	0.4101	0.5184	0.4259
	F1-Score	0.4365	0.5093	0.4445
ph	Best k	18	20	13
	CPU time (sec)	0.1906	3.9907	2.7960
	Accuracy	0.8056	0.7926	0.8019
	Average Precision	0.7685	0.7519	0.7678
	Average Recall	0.7574	0.7473	0.7386
	F1-Score	0.7625	0.7495	0.7500
ecl	Best k	6	17	3
	CPU time (sec)	0.0011	0.0168	0.0050
	Accuracy	0.8358	0.8507	0.8209
	Average Precision	0.7671	0.8955	0.8449
	Average Recall	0.7084	0.8317	0.7973
	F1-Score	0.7178	0.8404	0.8138
txr	Best k	13	19	4
	CPU time (sec)	0.6857	13.8499	4.3011
	Accuracy	0.9718	0.9736	0.9764
	Average Precision	0.9724	0.9739	0.9769
	Average Recall	0.9727	0.9737	0.9771
	F1-Score	0.9723	0.9737	0.9769
pd	Best k	12	20	4
	CPU time (sec)	1.5207	31.9614	6.4852
	Accuracy	0.9832	0.9754	0.9823
	Average Precision	0.9835	0.9758	0.9826
	Average Recall	0.9835	0.9762	0.9828
	F1-Score	0.9834	0.9760	0.9826
mgt	Best k	20	20	18
	CPU time (sec)	3.8574	75.7116	70.6558
	Accuracy	0.7960	0.7547	0.8073
	Average Precision	0.7813	0.7326	0.8131
	Average Recall	0.7640	0.7349	0.7566
	F1-Score	0.7708	0.7337	0.7719
sh	Best k	17	17	17
	CPU time (sec)	93.1168	582.4421	585.7447
	Accuracy	0.9971	0.9756	0.9962
	Average Precision	0.6214	0.5016	0.5449
	Average Recall	0.5594	0.8900	0.4509
	F1-Score	0.5807	0.5470	0.4648

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.14 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
tn	Best k	20	19	13
	CPU time (sec)	0.9311	18.8979	10.6151
	Accuracy	0.9311	0.8101	0.9743
	Average Precision	0.9312	0.8102	0.9743
	Average Recall	0.9311	0.8101	0.9743
	F1-Score	0.9311	0.8101	0.9743
pm	Best k	19	12	4
	CPU time (sec)	0.0037	0.0410	0.0150
	Accuracy	0.6667	0.6797	0.6863
	Average Precision	0.6289	0.6589	0.6573
	Average Recall	0.6253	0.6708	0.6625
	F1-Score	0.6268	0.6610	0.6594
lir	Best k	10	19	4
	CPU time (sec)	16.1777	294.8827	86.9005
	Accuracy	0.9343	0.9318	0.9215
	Average Precision	0.9339	0.9306	0.9210
	Average Recall	0.9336	0.9305	0.9208
	F1-Score	0.9331	0.9301	0.9203

Τα αποτελέσματα με βέλτιστο k για δεδομένα με θόρυβο 30% παρουσιάζονται στον Πίνακα 6.14. Ο Symmetric KNN επιτυγχάνει την υψηλότερη ακρίβεια σε έξι datasets: tn (4.32% υψηλότερη από τον Conventional), bn (6.04% υψηλότερη), bl (2.40% υψηλότερη), txr (0.46% υψηλότερη), mgt (1.13% υψηλότερη), και pm (1.96% υψηλότερη). Ο Conventional KNN υπερτερεί σε πέντε datasets: sh (0.09% υψηλότερη), ys (0.73% υψηλότερη), ph (0.37% υψηλότερη), pd (0.09% υψηλότερη), και lir (1.28% υψηλότερη), ενώ στο iris οι δύο μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Ο Mutual KNN επιτυγχάνει την καλύτερη απόδοση σε δύο datasets: ecl (1.49% υψηλότερη από τον Conventional) και ls (0.08% χαμηλότερη από τον Conventional αλλά 3.19% υψηλότερη από τον Symmetric). Αξιοσημείωτες είναι οι εντυπωσιακές βελτιώσεις του Symmetric KNN σε datasets με υψηλό θόρυβο όπως το tn (4.32% απόλυτη βελτίωση) και το bn (6.04% απόλυτη βελτίωση), υποδεικνύοντας την αποτελεσματικότητά του στο φιλτράρισμα θορυβωδών γειτόνων. Ο Symmetric KNN τείνει να προτιμά χαμηλότερες τιμές του k σε σχέση με τις άλλες δύο μεθόδους, ιδιαίτερα σε datasets όπως το txr, pd, pm, lir, και ecl. Ο Conventional KNN παραμένει ο ταχύτερος, με τον Mutual KNN να παρουσιάζει σημαντικά αυξημένους χρόνους εκτέλεσης, ειδικά στα μεγάλα datasets όπως το lir (18.2 φορές αργότερος) και το sh (6.3 φορές αργότερος).

Πίνακας 6.15: Αποτελέσματα Adaptive KNN με $k_{\max} = 9$ - Δεδομένα με θόρυβο 30%

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
bl	CPU time (sec)	0.0068	0.0187	0.0234
	Accuracy	0.720000	0.720000	0.768000
	Average Precision	0.563368	0.563368	0.591239
	Average Recall	0.547884	0.547884	0.582987
	F1-Score	0.553843	0.553843	0.586345
ph	CPU time (sec)	0.3001	2.2032	1.6713
	Accuracy	0.751852	0.735185	0.756481
	Average Precision	0.704317	0.688703	0.710215
	Average Recall	0.709295	0.701033	0.717083
	F1-Score	0.706636	0.693570	0.713334
txr	CPU time (sec)	3.5146	11.9363	9.0021
	Accuracy	0.967273	0.920000	0.960909
	Average Precision	0.966886	0.921223	0.960581
	Average Recall	0.967915	0.920202	0.961358
	F1-Score	0.967217	0.920378	0.960799
iris	CPU time (sec)	0.0010	0.0036	0.0021

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.15 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Accuracy	0.900000	0.833333	0.900000
	Average Precision	0.906061	0.837121	0.902357
	Average Recall	0.900000	0.833333	0.900000
	F1-Score	0.901504	0.830688	0.899749
	CPU time (sec)	3.3162	10.7259	7.8624
	Accuracy	0.798649	0.746622	0.818243
tn	Average Precision	0.798658	0.746643	0.818740
	Average Recall	0.798645	0.746628	0.818216
	F1-Score	0.798645	0.746619	0.818163
	CPU time (sec)	0.0069	0.0602	0.0543
	Accuracy	0.692810	0.692810	0.692810
pm	Average Precision	0.675026	0.681067	0.669798
	Average Recall	0.689623	0.698491	0.680755
	F1-Score	0.676853	0.680528	0.672616
	CPU time (sec)	0.0025	0.0162	0.0098
	Accuracy	0.820896	0.820896	0.865672
ecl	Average Precision	0.596218	0.717517	0.718400
	Average Recall	0.566667	0.692361	0.690046
	F1-Score	0.578898	0.699830	0.701080
	CPU time (sec)	0.2238	1.4866	1.0716
	Accuracy	0.722642	0.720755	0.728302
bn	Average Precision	0.721447	0.719676	0.727756
	Average Recall	0.722973	0.721240	0.729521
	F1-Score	0.721627	0.719796	0.727595
	CPU time (sec)	0.0274	0.2283	0.1977
	Accuracy	0.543919	0.486486	0.530405
ys	Average Precision	0.498991	0.384391	0.467985
	Average Recall	0.485153	0.478034	0.487587
	F1-Score	0.489696	0.400532	0.473929
	CPU time (sec)	4.3715	16.3742	12.4958
	Accuracy	0.871795	0.798757	0.867133
ls	Average Precision	0.860985	0.777326	0.856861
	Average Recall	0.854226	0.787476	0.848298
	F1-Score	0.856857	0.781508	0.851468
	CPU time (sec)	5.7809	19.3922	17.3114
	Accuracy	0.966333	0.918107	0.962693
pd	Average Precision	0.966304	0.917918	0.963353
	Average Recall	0.966546	0.918927	0.963413
	F1-Score	0.966341	0.918146	0.963270
	CPU time (sec)	11.4680	39.6948	32.2659
	Accuracy	0.705573	0.680074	0.710568
mgt	Average Precision	0.682198	0.657878	0.684734
	Average Recall	0.688661	0.665729	0.686231
	F1-Score	0.684643	0.660152	0.685447
	CPU time (sec)	20.7575	65.5290	55.5025
	Accuracy	0.919500	0.877250	0.890750
lir	Average Precision	0.918974	0.876647	0.891082
	Average Recall	0.918836	0.876672	0.890157
	F1-Score	0.918445	0.876240	0.889742
	CPU time (sec)	109.8184	375.0546	352.3461
	Accuracy	0.969049	0.922321	0.948185
sh			Συνεχίζεται στην επόμενη σελίδα	

Πίνακας 6.15 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Average Precision	0.488745	0.426772	0.450116
	Average Recall	0.796912	0.867334	0.858953
	F1-Score	0.526681	0.455881	0.483886

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = 9$ σε δεδομένα με θόρυβο 30% παρουσιάζονται στον Πίνακα 6.15. Ο Conventional KNN επιτυγχάνει την υψηλότερη ακρίβεια σε επτά datasets (sh, pd, txr, lir, ys, ls, iris), με τις πιο σημαντικές υπεροχές να εμφανίζονται στο sh (2.09% υψηλότερη ακρίβεια από τον Symmetric), pd (0.36% υψηλότερη), και txr (0.64% υψηλότερη). Ο Symmetric KNN υπερτερεί σε πέντε datasets: ecl (4.48% υψηλότερη ακρίβεια), bl (4.80% υψηλότερη), tn (1.96% υψηλότερη), bn (0.56% υψηλότερη), ph (0.46% υψηλότερη), και mgt (0.50% υψηλότερη). Ο Mutual KNN παρουσιάζει τη χειρότερη απόδοση σε όλα τα datasets, με σημαντικές αποκλίσεις στο pd (4.83% χαμηλότερη ακρίβεια από τον Conventional), txr (4.73% χαμηλότερη), και ls (7.30% χαμηλότερη). Στο pm οι τρεις μέθοδοι επιτυγχάνουν την ίδια ακρίβεια. Η adaptive μέθοδος με $k_{\max} = 9$ φαίνεται να ευνοεί τον Conventional KNN σε datasets με θόρυβο, ενώ ο Symmetric KNN διατηρεί την ικανότητά του να φιλτράρει θόρυβο σε συγκεκριμένα datasets όπως το ecl και το bl. Ο Conventional KNN παραμένει ο ταχύτερος, με τον Mutual KNN να απαιτεί 3-7 φορές περισσότερο χρόνο στα μεγάλα datasets.

Πίνακας 6.16: Αποτελέσματα Adaptive KNN με $k_{\max} = 20$ - Δεδομένα με θόρυβο 30%

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	0.0013	0.0095	0.0072
	Accuracy	0.933333	0.933333	0.866667
iris	Average Precision	0.933333	0.936364	0.866667
	Average Recall	0.933333	0.933333	0.866667
	F1-Score	0.933333	0.933250	0.866667
	CPU time (sec)	0.0038	0.0212	0.0278
	Accuracy	0.776119	0.820896	0.820896
ecl	Average Precision	0.574603	0.824780	0.690972
	Average Recall	0.476488	0.771528	0.627778
	F1-Score	0.486652	0.788064	0.654091
	CPU time (sec)	0.0079	0.0250	0.0465
	Accuracy	0.752000	0.744000	0.768000
bl	Average Precision	0.545482	0.540135	0.591239
	Average Recall	0.545482	0.539072	0.582987
	F1-Score	0.545482	0.539561	0.586345
	CPU time (sec)	2.9962	23.6504	16.3563
	Accuracy	0.884227	0.845377	0.864802
ls	Average Precision	0.873318	0.828901	0.855643
	Average Recall	0.866495	0.831544	0.844032
	F1-Score	0.869069	0.829331	0.848341
	CPU time (sec)	3.4115	13.7334	9.2469
	Accuracy	0.822973	0.778378	0.819595
tn	Average Precision	0.823092	0.778380	0.819966
	Average Recall	0.822960	0.778376	0.819571
	F1-Score	0.822952	0.778377	0.819535
	CPU time (sec)	0.0392	0.4025	0.3192
	Accuracy	0.570946	0.520270	0.506757
ys	Average Precision	0.530010	0.449781	0.455677
	Average Recall	0.502807	0.500330	0.464663
	F1-Score	0.513134	0.463766	0.457555
	CPU time (sec)	0.0115	0.0846	0.0559
	Accuracy	0.686275	0.673203	0.686275
pm	Average Precision	0.664286	0.659679	0.661796

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.16 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Average Recall	0.675755	0.674623	0.671321
	F1-Score	0.666788	0.659212	0.664474
	CPU time (sec)	0.3564	2.0533	1.3167
	Accuracy	0.745283	0.743396	0.730189
bn	Average Precision	0.744887	0.742455	0.729791
	Average Recall	0.746806	0.744229	0.731592
	F1-Score	0.744669	0.742571	0.729538
	CPU time (sec)	18.1143	103.1511	79.6029
	Accuracy	0.919500	0.916750	0.874250
lir	Average Precision	0.919256	0.916264	0.875643
	Average Recall	0.918825	0.915924	0.873602
	F1-Score	0.918389	0.915732	0.873528
	CPU time (sec)	1.5655	2.7754	2.0825
	Accuracy	0.762963	0.759259	0.754630
ph	Average Precision	0.717466	0.713728	0.708280
	Average Recall	0.723493	0.721756	0.715766
	F1-Score	0.720251	0.717318	0.711640
	CPU time (sec)	2.4540	11.3885	11.5145
	Accuracy	0.970000	0.962727	0.954545
txr	Average Precision	0.970237	0.963033	0.954588
	Average Recall	0.970487	0.962686	0.954631
	F1-Score	0.970233	0.962795	0.954356
	CPU time (sec)	3.7296	33.0685	22.4994
	Accuracy	0.970883	0.961783	0.959054
pd	Average Precision	0.971291	0.962245	0.959704
	Average Recall	0.971475	0.962109	0.959874
	F1-Score	0.971275	0.962129	0.959647
	CPU time (sec)	12.7998	59.7287	38.3927
	Accuracy	0.718980	0.708991	0.711094
mgt	Average Precision	0.694576	0.685766	0.685207
	Average Recall	0.698066	0.692308	0.686473
	F1-Score	0.696136	0.688258	0.685815
	CPU time (sec)	114.8310	461.4175	381.6232
	Accuracy	0.975688	0.960255	0.949306
sh	Average Precision	0.505700	0.473142	0.448213
	Average Recall	0.790739	0.883142	0.843214
	F1-Score	0.543117	0.513687	0.481022

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = 20$ σε δεδομένα με θόρυβο 30% παρουσιάζονται στον Πίνακα 6.16. Ο Conventional KNN επιτυγχάνει την υψηλότερη ακρίβεια σε εννέα datasets (sh, pd, txr, lir, ph, bn, ys, ls, iris), με τις πιο σημαντικές υπεροχές να εμφανίζονται στο sh (1.54% υψηλότερη ακρίβεια από τον Mutual και 2.64% υψηλότερη από τον Symmetric), pd (0.91% υψηλότερη από τον Mutual), και txr (0.73% υψηλότερη από τον Mutual). Ο Mutual KNN υπερτερεί μόνο στο ecl (4.48% υψηλότερη ακρίβεια από τον Conventional), ενώ ο Symmetric KNN επιτυγχάνει την καλύτερη απόδοση σε τρία datasets: bl (1.60% υψηλότερη από τον Conventional), pm (ισοπαλία με Conventional), και ecl (ισοπαλία με Mutual). Αξιοσημείωτη είναι η σημαντική υποβάθμιση του Symmetric KNN στο iris (6.67% χαμηλότερη ακρίβεια) και στο lir (4.53% χαμηλότερη), υποδεικνύοντας ότι το μεγαλύτερο $k_{\max} = 20$ δεν ευνοεί πάντα τη Symmetric μέθοδο παρουσία θορύβου. Ο Conventional KNN διατηρεί τη σταθερότητά του σε μεγαλύτερο εύρος γειτόνων, ενώ ο Mutual KNN παρουσιάζει μεγάλη υπολογιστική επιβάρυνση, απαιτώντας 4-6 φορές περισσότερο χρόνο στα μεγάλα datasets.

Πίνακας 6.17: Αποτελέσματα Adaptive KNN με $k_{\max} = 50$ - Δεδομένα με θόρυβο 30%

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	0.0023	0.0274	0.0557
	Accuracy	0.776119	0.820896	0.791045
ecl	Average Precision	0.601544	0.800980	0.699429
	Average Recall	0.472024	0.738194	0.601736
	F1-Score	0.495116	0.755335	0.639568
	CPU time (sec)	0.0123	0.0341	0.0749
	Accuracy	0.752000	0.752000	0.768000
bl	Average Precision	0.545482	0.545482	0.591239
	Average Recall	0.545482	0.545482	0.582987
	F1-Score	0.545482	0.545482	0.586345
	CPU time (sec)	0.0178	0.0835	0.0595
	Accuracy	0.705882	0.712418	0.686275
pm	Average Precision	0.679220	0.693706	0.661796
	Average Recall	0.686321	0.709057	0.671321
	F1-Score	0.681913	0.696538	0.664474
	CPU time (sec)	0.0008	0.0069	0.0067
	Accuracy	0.900000	0.933333	0.866667
iris	Average Precision	0.902357	0.944444	0.866667
	Average Recall	0.900000	0.933333	0.866667
	F1-Score	0.899749	0.932660	0.866667
	CPU time (sec)	0.0231	0.6466	0.6443
	Accuracy	0.567568	0.547297	0.513514
ys	Average Precision	0.521187	0.507465	0.454190
	Average Recall	0.490537	0.510680	0.444477
	F1-Score	0.502073	0.504749	0.444655
	CPU time (sec)	0.7081	3.8935	2.9541
	Accuracy	0.763889	0.762037	0.755556
ph	Average Precision	0.718000	0.716302	0.709456
	Average Recall	0.721455	0.721935	0.717324
	F1-Score	0.719654	0.718918	0.712970
	CPU time (sec)	0.4614	2.9193	2.3453
	Accuracy	0.746226	0.741509	0.731132
bn	Average Precision	0.745477	0.740334	0.730811
	Average Recall	0.747335	0.741990	0.732627
	F1-Score	0.745490	0.740564	0.730508
	CPU time (sec)	3.3515	23.9550	21.7475
	Accuracy	0.967273	0.968182	0.946364
txr	Average Precision	0.967722	0.968713	0.946501
	Average Recall	0.967528	0.968616	0.946411
	F1-Score	0.967428	0.968572	0.945936
	CPU time (sec)	2.4641	33.1970	33.9611
	Accuracy	0.878788	0.874903	0.860917
ls	Average Precision	0.871303	0.861949	0.850845
	Average Recall	0.859442	0.857284	0.840101
	F1-Score	0.864195	0.858657	0.843938
	CPU time (sec)	3.1946	17.7363	18.7773
	Accuracy	0.838514	0.808784	0.820270
tn	Average Precision	0.838566	0.808789	0.820736
	Average Recall	0.838505	0.808781	0.820244

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.17 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	F1-Score	0.838505	0.808782	0.820196
mgt	CPU time (sec)	13.4398	82.1556	56.3953
	Accuracy	0.726604	0.724501	0.711619
	Average Precision	0.702469	0.701049	0.685777
	Average Recall	0.705306	0.705998	0.687047
	F1-Score	0.703776	0.703158	0.686387
lir	CPU time (sec)	20.8530	181.1314	133.0649
	Accuracy	0.906000	0.910250	0.845750
	Average Precision	0.906270	0.909938	0.850028
	Average Recall	0.905486	0.909062	0.845265
	F1-Score	0.905170	0.909063	0.845526
pd	CPU time (sec)	2.8583	49.3591	42.8911
	Accuracy	0.967243	0.969518	0.945860
	Average Precision	0.967731	0.969867	0.947804
	Average Recall	0.967921	0.969637	0.947278
	F1-Score	0.967704	0.969684	0.947287
sh	CPU time (sec)	111.7000	607.7585	488.8690
	Accuracy	0.974998	0.972153	0.949047
	Average Precision	0.501497	0.499660	0.448381
	Average Recall	0.773956	0.857448	0.843039
	F1-Score	0.537080	0.539543	0.481222

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = 50$ σε δεδομένα με θόρυβο 30% παρουσιάζονται στον Πίνακα 6.17. Ο Conventional KNN επιτυγχάνει την υψηλότερη ακρίβεια σε οκτώ datasets (sh, tn, bn, ls, ys, ph, mgt, και ισοπαλία με Mutual στο bl), με τις πιο σημαντικές υπεροχές να εμφανίζονται στο sh (0.28% υψηλότερη ακρίβεια από τον Mutual και 2.60% υψηλότερη από τον Symmetric) και στο tn (2.97% υψηλότερη από τον Mutual). Ο Mutual KNN υπερτερεί σε τέσσερα datasets: lir (0.43% υψηλότερη ακρίβεια), pd (0.23% υψηλότερη), pm (0.65% υψηλότερη), txr (0.09% υψηλότερη), ecl (4.48% υψηλότερη), και iris (3.33% υψηλότερη). Ο Symmetric KNN επιτυγχάνει την καλύτερη απόδοση μόνο στο bl (1.60% υψηλότερη από τον Conventional), ενώ παρουσιάζει σημαντική υποβάθμιση στα lir (6.03% χαμηλότερη), pd (2.14% χαμηλότερη), και txr (2.09% χαμηλότερη). Η αύξηση του $k_{\max}$ στο 50 φαίνεται να ευνοεί τον Mutual KNN σε συγκεκριμένα datasets όπως το lir, pd, και ecl, ενώ ο Conventional KNN διατηρεί τη σταθερότητά του στα περισσότερα datasets. Ο Mutual KNN παρουσιάζει σημαντική υπολογιστική επιβάρυνση, απαιτώντας 5-17 φορές περισσότερο χρόνο στα μεγάλα datasets, με ιδιαίτερα εντυπωσιακή την αύξηση στο lir (8.7 φορές αργότερος).

Πίνακας 6.18: Αποτελέσματα Adaptive KNN $k_{\max} = \sqrt{n/2}$ - Δεδομένα με θόρυβο 30%

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
iris	CPU time (sec)	0.017	0.079	0.063
	Accuracy	0.9000	0.9000	0.9000
	Average Precision	0.9061	0.9061	0.9024
	Average Recall	0.9000	0.9000	0.9000
	F1-Score	0.9015	0.9015	0.8997
ecl	CPU time (sec)	0.071	0.719	0.372
	Accuracy	0.7910	0.7910	0.8209
	Average Precision	0.8014	0.8014	0.8221
	Average Recall	0.7910	0.7910	0.8209
	F1-Score	0.7753	0.7753	0.8095
bl	CPU time (sec)	0.225	1.541	1.299
	Accuracy	0.7440	0.7440	0.7360
	Average Precision	0.7506	0.7506	0.7485
	Average Recall	0.7440	0.7440	0.7360

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.18 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	F1-Score	0.7473	0.7473	0.7419
	CPU time (sec)	0.351	3.307	1.694
	Accuracy	0.6863	0.6863	0.7190
pm	Average Precision	0.7026	0.7026	0.7292
	Average Recall	0.6863	0.6863	0.7190
	F1-Score	0.6915	0.6915	0.7226
	CPU time (sec)	17.71	217.64	140.66
	Accuracy	0.7453	0.7358	0.7094
bn	Average Precision	0.7484	0.7399	0.7118
	Average Recall	0.7453	0.7358	0.7094
	F1-Score	0.7458	0.7364	0.7099
	CPU time (sec)	27.78	555.51	414.69
	Accuracy	0.8765	0.8765	0.8407
ls	Average Precision	0.8754	0.8754	0.8407
	Average Recall	0.8765	0.8765	0.8407
	F1-Score	0.8751	0.8751	0.8394
	CPU time (sec)	1.347	23.28	15.23
	Accuracy	0.5709	0.5574	0.5101
ys	Average Precision	0.5605	0.5533	0.5055
	Average Recall	0.5709	0.5574	0.5101
	F1-Score	0.5624	0.5519	0.5057
	CPU time (sec)	18.65	234.91	147.48
	Accuracy	0.7639	0.7472	0.7463
ph	Average Precision	0.7654	0.7523	0.7558
	Average Recall	0.7639	0.7472	0.7463
	F1-Score	0.7646	0.7495	0.7501
	CPU time (sec)	20.65	346.15	278.50
	Accuracy	0.9673	0.9564	0.9191
txr	Average Precision	0.9678	0.9569	0.9194
	Average Recall	0.9673	0.9564	0.9191
	F1-Score	0.9674	0.9564	0.9189
	CPU time (sec)	33.90	543.01	402.29
	Accuracy	0.8399	0.8399	0.8041
tn	Average Precision	0.8399	0.8399	0.8044
	Average Recall	0.8399	0.8399	0.8041
	F1-Score	0.8399	0.8399	0.8040
	CPU time (sec)	240.36	3638.66	2574.19
	Accuracy	0.7271	0.7140	0.6990
mgt	Average Precision	0.7286	0.7173	0.7027
	Average Recall	0.7271	0.7140	0.6990
	F1-Score	0.7278	0.7154	0.7006
	CPU time (sec)	78.36	1700.08	1415.40
	Accuracy	0.9654	0.9654	0.9167
pd	Average Precision	0.9656	0.9656	0.9177
	Average Recall	0.9654	0.9654	0.9167
	F1-Score	0.9654	0.9654	0.9162
	CPU time (sec)	277.18	5971.95	4920.90
	Accuracy	0.8970	0.8168	0.8045
lir	Average Precision	0.8996	0.8219	0.8126
	Average Recall	0.8970	0.8168	0.8045
	F1-Score	0.8974	0.8173	0.8056

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 6.18 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	1975.88	97286.20	93893.27
	Accuracy	0.9749	0.9749	0.9749
sh	Average Precision	0.9864	0.9864	0.9864
	Average Recall	0.9749	0.9749	0.9749
	F1-Score	0.9802	0.9802	0.9802

Τα αποτελέσματα με τη μέθοδο adaptive KNN και $k_{\max} = \sqrt{n/2}$ σε δεδομένα με θόρυβο 30% παρουσιάζονται στον Πίνακα 6.18. Ο Conventional KNN επιτυγχάνει την υψηλότερη ακρίβεια σε έξι datasets (lir, mgt, txr, ph, ys, bn), με τις πιο σημαντικές υπεροχές να εμφανίζονται στο lir (8.02% υψηλότερη ακρίβεια από τον Mutual και 9.25% υψηλότερη από τον Symmetric), txr (1.09% υψηλότερη από τον Mutual και 4.82% υψηλότερη από τον Symmetric), και ph (1.67% υψηλότερη από τον Mutual). Ο Symmetric KNN επιτυγχάνει την καλύτερη απόδοση σε δύο datasets: ecl (2.99% υψηλότερη ακρίβεια) και pm (3.27% υψηλότερη). Στα υπόλοιπα datasets (iris, sh, tn, ls, pd, bl) οι μέθοδοι παρουσιάζουν ίδια ή πολύ παρόμοια ακρίβεια. Αξιοσημείωτη είναι η εξαιρετική υπολογιστική επιβάρυνση όλων των μεθόδων με το $k_{\max} = \sqrt{n/2}$, ιδιαίτερα στα μεγάλα datasets: ο Mutual KNN απαιτεί έως 49 φορές περισσότερο χρόνο από τον Conventional στο sh, 22 φορές στο pd, και 21 φορές στο lir. Η χρήση του δυναμικού $k_{\max} = \sqrt{n/2}$ οδηγεί σε πολύ μεγάλες τιμές του k για μεγάλα datasets, με αποτέλεσμα δραματική αύξηση του υπολογιστικού κόστους χωρίς ανάλογη βελτίωση της ακρίβειας.

Πίνακας 6.19: Αποτελέσματα Adaptive KNN $k_{\max} = 20$ Stable LastK (w=3) - Δεδομένα με θόρυβο 30%

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	CPU time (sec)	0.23	5.61	5.84
	Accuracy	0.8640	0.8640	0.8880
bl	Average Precision	0.8055	0.8055	0.8203
	Average Recall	0.8640	0.8640	0.8880
	F1-Score	0.8319	0.8319	0.8517
	CPU time (sec)	19.80	430.14	404.85
	Accuracy	0.7943	0.7792	0.8368
bn	Average Precision	0.7954	0.7807	0.8367
	Average Recall	0.7943	0.7792	0.8368
	F1-Score	0.7946	0.7796	0.8367
	CPU time (sec)	0.06	1.89	1.43
	Accuracy	0.8209	0.8209	0.7612
ecl	Average Precision	0.8209	0.8209	0.7698
	Average Recall	0.8209	0.8209	0.7612
	F1-Score	0.8094	0.8094	0.7436
	CPU time (sec)	0.01	0.49	0.32
	Accuracy	0.9667	0.9667	0.9333
iris	Average Precision	0.9697	0.9697	0.9444
	Average Recall	0.9667	0.9667	0.9333
	F1-Score	0.9666	0.9666	0.9327
	CPU time (sec)	297.06	5188.69	4865.19
	Accuracy	0.9235	0.8223	0.8650
lir	Average Precision	0.9260	0.8265	0.8713
	Average Recall	0.9235	0.8223	0.8650
	F1-Score	0.9239	0.8224	0.8661
	CPU time (sec)	32.83	701.48	634.66
	Accuracy	0.8990	0.8990	0.8718
ls	Average Precision	0.8971	0.8971	0.8721
	Average Recall	0.8990	0.8990	0.8718
	F1-Score	0.8971	0.8971	0.8692
	CPU time (sec)	254.42	4688.26	4331.56

Συνεχίζεται στην επόμενη σελίδα

mgt

Πίνακας 6.19 – Συνέχεια από προηγούμενη σελίδα

Dataset	Μετρική	Conventional KNN	Mutual KNN	Symmetric KNN
	Accuracy	0.7744	0.7634	0.7992
	Average Precision	0.7714	0.7606	0.7969
	Average Recall	0.7744	0.7634	0.7992
	F1-Score	0.7724	0.7617	0.7926
	CPU time (sec)	86.67	1954.94	1571.43
	Accuracy	0.9791	0.9791	0.9709
pd	Average Precision	0.9794	0.9794	0.9716
	Average Recall	0.9791	0.9791	0.9709
	F1-Score	0.9790	0.9790	0.9708
	CPU time (sec)	19.60	456.08	413.41
	Accuracy	0.8037	0.7667	0.8102
ph	Average Precision	0.8011	0.7667	0.8053
	Average Recall	0.8037	0.7667	0.8102
	F1-Score	0.8022	0.7667	0.8066
	CPU time (sec)	0.47	8.69	9.65
	Accuracy	0.6863	0.6863	0.6993
pm	Average Precision	0.6891	0.6891	0.6871
	Average Recall	0.6863	0.6863	0.6993
	F1-Score	0.6876	0.6876	0.6887
	CPU time (sec)	2339.43	98176.20	94383.27
	Accuracy	0.9897	0.9898	0.9909
sh	Average Precision	0.9933	0.9935	0.9940
	Average Recall	0.9897	0.9898	0.9909
	F1-Score	0.9914	0.9916	0.9923
	CPU time (sec)	37.95	890.32	833.51
	Accuracy	0.9081	0.9081	0.9588
tn	Average Precision	0.9082	0.9082	0.9588
	Average Recall	0.9081	0.9081	0.9588
	F1-Score	0.9081	0.9081	0.9588
	CPU time (sec)	23.76	507.41	463.02
	Accuracy	0.9682	0.9464	0.9536
txr	Average Precision	0.9686	0.9477	0.9549
	Average Recall	0.9682	0.9464	0.9536
	F1-Score	0.9683	0.9462	0.9537
	CPU time (sec)	1.24	36.77	33.09
	Accuracy	0.5811	0.5574	0.5507
ys	Average Precision	0.5767	0.5632	0.5405
	Average Recall	0.5811	0.5574	0.5507
	F1-Score	0.5736	0.5561	0.5418

Τα αποτελέσματα με τη μέθοδο adaptive KNN με $k_{\max} = 20$, stable lastk (w=3) σε δεδομένα με θόρυβο 30% παρουσιάζονται στον Πίνακα 6.19. Ο Symmetric KNN επιτυγχάνει εντυπωσιακή υπεροχή σε έξι datasets: tn (5.07% υψηλότερη ακρίβεια από τον Conventional), bn (4.25% υψηλότερη), mgt (2.48% υψηλότερη), bl (2.40% υψηλότερη), pm (1.30% υψηλότερη), και ph (0.65% υψηλότερη). Ο Conventional KNN υπερτερεί σε πέντε datasets: lir (10.12% υψηλότερη ακρίβεια), txr (1.46% υψηλότερη), ys (2.37% υψηλότερη), ecl (5.97% υψηλότερη), και ls (2.72% υψηλότερη). Στα iris, pd και sh οι μέθοδοι Conventional και Mutual επιτυγχάνουν παρόμοια ή ταυτόσημη ακρίβεια, με τον Symmetric να υπολείπεται ελαφρώς. Ο Mutual KNN εμφανίζει την καλύτερη απόδοση μόνο στο sh (0.01% υψηλότερη από τον Conventional και 0.11% χαμηλότερη από τον Symmetric). Αξιοσημείωτη είναι η εξαιρετική βελτίωση του Symmetric KNN στο tn (5.07% απόλυτη βελτίωση) παρουσία θορύβου με τη χρήση του stable lastk window, υποδεικνύοντας την αποτελεσματικότητα της σταθεροποίησης του k στο φιλτράρισμα θορύβου. Ωστόσο, η μέθοδος παρουσιάζει δραματική υπολογιστική επιβάρυνση, με τον Mutual KNN να απαιτεί 17-42 φορές περισσότερο χρόνο στα μεγάλα datasets.

6.4 Στατιστικά πειράματα

6.4.1 Friedman Test

Το Friedman test χρησιμοποιήθηκε για να εξετάσει αν υπάρχουν στατιστικά σημαντικές διαφορές μεταξύ των μεθόδων σε όλα τα datasets.

Αποτελέσματα χωρίς θόρυβο

Πίνακας 6.20: Friedman Test - Δεδομένα χωρίς θόρυβο

Μέτρηση	**Τιμή**
Friedman statistic	96.909
P-value	4.16814E-10
Degrees of freedom	26
Αποτέλεσμα	**Στατιστικά σημαντικό** ($p < 0.05$)

Πίνακας 6.21: Κατάταξη μεθόδων (χωρίς θόρυβο) - Χαμηλότερη κατάταξη = Καλύτερη απόδοση

Μέθοδος	**Μέση Κατάταξη**	**Μέση Ακρίβεια**
Mutual Adaptive kmax20 lastk w3	6.39	0.8999
Mutual k=bestk	6.68	0.8968
Custom k=bestk	6.71	0.8967
Mutual k=9	9.64	0.8920
Mutual Adaptive kmax20	10.04	0.8888
Custom Adaptive kmax20 lastk w3	11.93	0.8965
Custom k=5	12.21	0.8902
Mutual Adaptive kmax50	12.39	0.8863
Mutual Adaptive kmax sqrt(n/2)	12.68	0.8855
Mutual Adaptive kmax9	12.96	0.8838
Custom k=9	13.00	0.8931
Custom Adaptive kmax20	13.04	0.8913
Custom Adaptive kmax9	13.07	0.8912
Custom Adaptive kmax50	13.54	0.8889
Mutual k=5	13.79	0.8831
Symmetric k=bestk	13.96	0.8891
Custom Adaptive kmax sqrt(n/2)	14.43	0.8876
Symmetric Adaptive kmax9	16.32	0.8850
Symmetric k=5	17.07	0.8841
Symmetric Adaptive kmax50	17.43	0.8829
Symmetric k=9	17.57	0.8781
Custom k=1	17.86	0.8679
Mutual k=1	17.86	0.8679
Symmetric Adaptive kmax sqrt(n/2)	18.07	0.8826
Symmetric Adaptive kmax20	18.14	0.8814
Symmetric Adaptive kmax20 lastk w3	18.54	0.8725
Symmetric k=1	22.68	0.8601

Αποτελέσματα με θόρυβο 30%

Πίνακας 6.22: Friedman Test - Δεδομένα με θόρυβο 30%

Μέτρηση	**Τιμή**
Friedman statistic	248.072
P-value	4.14708E-38
Degrees of freedom	26
Αποτέλεσμα	**Στατιστικά σημαντικό** ($p < 0.05$)

Πίνακας 6.23: Κατάταξη μεθόδων με θόρυβο 30% - Χαμηλότερη κατάταξη = Καλύτερη απόδοση

Μέθοδος	**Μέση Κατάταξη**	**Μέση Ακρίβεια**
Conventional KNN k=bestk	4.50	0.8634
Symmetric KNN k=bestk	4.61	0.8690
Conventional KNN Adaptive kmax20 lastk w3	6.04	0.8542
Symmetric KNN k=5	6.21	0.8523
Symmetric KNN k=9	6.29	0.8621
Conventional KNN k=9	7.57	0.8358
Symmetric KNN Adaptive kmax20 lastk w3	8.50	0.8488
Mutual KNN k=bestk	9.04	0.8373
Mutual KNN Adaptive kmax20 lastk w3	10.50	0.8327
Conventional KNN Adaptive kmax20	10.79	0.8207
Conventional KNN Adaptive kmax50	11.68	0.8194
Mutual KNN Adaptive kmax50	11.71	0.8213
Conventional KNN Adaptive kmax sqrt(n/2)	12.68	0.8178
Mutual KNN Adaptive kmax sqrt(n/2)	14.07	0.8104
Symmetric KNN Adaptive kmax9	14.71	0.8143
Conventional KNN Adaptive kmax9	15.07	0.8107
Symmetric KNN Adaptive kmax sqrt(n/2)	16.11	0.8047
Mutual KNN Adaptive kmax20	16.36	0.8092
Symmetric KNN Adaptive kmax20	16.89	0.8047
Symmetric KNN Adaptive kmax50	17.50	0.7994
Mutual KNN k=9	17.89	0.7940
Conventional KNN k=5	18.29	0.7978
Mutual KNN Adaptive kmax9	20.64	0.7766
Mutual KNN k=5	22.36	0.7372
Mutual KNN k=1	25.89	0.6366
Conventional KNN k=1	25.89	0.6366
Symmetric KNN k=1	26.21	0.6375

6.4.2 Wilcoxon Signed-Rank Test

Το Wilcoxon signed-rank test χρησιμοποιήθηκε για pairwise συγκρίσεις μεταξύ του custom kNN, mutual kNN και symmetric kNN. Τα αποτελέσματα παρουσιάζονται ως: (νικητής custom vs mutual, νικητής custom vs symmetric, στατιστική σημαντικότητα custom vs mutual, στατιστική σημαντικότητα custom vs symmetric).

Αποτελέσματα χωρίς θόρυβο

Πίνακας 6.24: Wilcoxon Test - Δεδομένα χωρίς θόρυβο

Μέθοδος	**Conventional vs Mutual**	**Conventional vs Symmetric**	**Σημ. C-M**	**Σημ. C-S**
k=1	Mutual KNN	Conventional KNN	Όχι	Όχι
k=5	Conventional KNN	**Conventional KNN**	Όχι	**Ναι**
k=9	Conventional KNN	Conventional KNN	Όχι	Όχι
k=best	Mutual KNN	**Conventional KNN**	Όχι	**Ναι**
Adaptive k=9	Conventional KNN	**Conventional KNN**	Όχι	**Ναι**
Adaptive k=20	Conventional KNN	**Conventional KNN**	Όχι	**Ναι**
Adaptive k=50	Conventional KNN	**Conventional KNN**	Όχι	**Ναι**
Adaptive k=sqrt(n/2)	Conventional KNN	**Conventional KNN**	Όχι	**Ναι**
Adaptive w3 lastk=20	Mutual KNN	**Conventional KNN**	Όχι	**Ναι**

Ερμηνεία: Οι στήλες "Conventional vs Mutual/Symmetric" δείχνουν ποια μέθοδος είχε καλύτερη απόδοση. Οι στήλες "Σημ." δείχνουν αν η διαφορά είναι στατιστικά σημαντική (Ναι = $p < 0.05$, Όχι = $p \geq 0.05$).

Αποτελέσματα με θόρυβο 30%

Πίνακας 6.25: Wilcoxon Test - Δεδομένα με θόρυβο 30%

Μέθοδος	**Conventional vs Mutual**	**Conventional vs Symmetric**	**Σημ. C-M**	**Σημ. C-S**
k=1	Mutual KNN	Symmetric KNN	Όχι	Όχι
k=5	**Conventional KNN**	**Symmetric KNN**	**Ναι**	**Ναι**
k=9	**Conventional KNN**	Symmetric KNN	**Ναι**	Όχι
k=best	**Conventional KNN**	Symmetric KNN	**Ναι**	Όχι
Adaptive k=9	**Conventional KNN**	Symmetric KNN	**Ναι**	Όχι
Adaptive k=20	**Conventional KNN**	**Conventional KNN**	**Ναι**	**Ναι**
Adaptive k=50	Mutual KNN	**Conventional KNN**	Όχι	**Ναι**
Adaptive k=sqrt(n/2)	Conventional KNN	Conventional KNN	Όχι	Όχι
Adaptive w3 lastk=20	**Conventional KNN**	Conventional KNN	**Ναι**	Όχι

Ερμηνεία: Οι στήλες "Conventional vs Mutual/Symmetric" δείχνουν ποια μέθοδος είχε καλύτερη απόδοση. Οι στήλες "Σημ." δείχνουν αν η διαφορά είναι στατιστικά σημαντική (Ναι = $p < 0.05$, Όχι = $p \geq 0.05$).

Κεφάλαιο 7

Συμπεράσματα

7.1 Συμπεράσματα Στατιστικών Πειραμάτων

Η παρούσα εργασία πραγματοποίησε εκτενή πειραματική αξιολόγηση των αλγορίθμων k-NN και των παραλλαγών τους σε 14 datasets από το UCI Machine Learning Repository και το KEEL Repository, εξετάζοντας τη συμπεριφορά τους τόσο σε καθαρά δεδομένα όσο και σε δεδομένα με θόρυβο 30%. Τα στατιστικά τεστ που διεξήχθησαν επιβεβαίωσαν την ύπαρξη σημαντικών διαφορών μεταξύ των μεθόδων και αποκάλυψαν ενδιαφέροντα μοτίβα απόδοσης. Το Friedman test κατέδειξε στατιστικά σημαντικές διαφορές τόσο στα καθαρά δεδομένα ($p = 4.17\times10^{-10}$) όσο και στα δεδομένα με θόρυβο 30% ($p = 4.15\times10^{-38}$), επιβεβαιώνοντας ότι η επιλογή της μεθόδου επηρεάζει ουσιαστικά την απόδοση ταξινόμησης. Η σύγκριση των τριών βασικών μεθόδων—Conventional, Mutual και Symmetric k-NN—αποκάλυψε διακριτά χαρακτηριστικά για κάθε προσέγγιση. Ο Conventional k-NN επέδειξε συνεπή και σταθερή απόδοση σε ευρύ φάσμα συνθηκών, αναδεικνυόμενος ως ο ταχύτερος σε όλα τα σενάρια και επιτυγχάνοντας υψηλή ακρίβεια ιδιαίτερα με βέλτιστη επιλογή του k ανά dataset. Ο Mutual k-NN, που βασίζεται σε αμοιβαίες γειτονικές σχέσεις, υπερτέρησε σε επτά datasets όταν χρησιμοποιήθηκε βέλτιστο k, επιδεικνύοντας ότι το φιλτράρισμα μέσω αμοιβαιότητας μπορεί να βελτιώσει την ακρίβεια σε δεδομένα με πολύπλοκη δομή, αν και με σημαντικά αυξημένο υπολογιστικό κόστος (2-30 φορές αργότερος από τον Conventional). Ο Symmetric k-NN παρουσίασε χαμηλότερη απόδοση σε καθαρά δεδομένα, αλλά αναδείχθηκε ως εξαιρετικά αποτελεσματικός παρουσία θορύβου, επιτυγχάνοντας την υψηλότερη ακρίβεια σε εννέα από τα δεκατέσσερα datasets για $k = 5$ με θόρυβο 30%, με εντυπωσιακές βελτιώσεις στο tn (+14.39%), bl (+9.6%), bn (+8.21%) και ys (+7.77%). Αυτή η εξαιρετική απόδοση αποδίδεται στην ικανότητα της συμμετρικής μεθόδου να φιλτράρει αποτελεσματικά θορυβώδεις γείτονες μέσω της απαίτησης αμοιβαίας συμφωνίας από δύο κατευθύνσεις.

Οι adaptive μέθοδοι που αναπτύχθηκαν στην εργασία—Ada Mutual k-NN και Ada Symmetric k-NN—αποτέλεσαν την πρώτη συστηματική προσπάθεια συνδυασμού της ιδέας του δυναμικού k με τις έννοιες της αμοιβαιότητας και της συμμετρίας. Τα αποτελέσματα έδειξαν ότι στα καθαρά δεδομένα, οι adaptive μέθοδοι του Mutual k-NN κατέλαβαν τις υψηλότερες θέσεις στο Friedman test, με τη μέθοδο lastk window=3 να επιτυγχάνει μέση κατάταξη 6.39 και σημαντικές βελτιώσεις σε datasets όπως το pd (+0.86%), txr (+1.55%) και lir (+1.80%). Η αύξηση του μέγιστου εύρους γειτόνων από $k_{max} = 9$ σε $k_{max} = 50$ γενικά ευνοούσε τον Mutual k-NN, με τη μέθοδο να κερδίζει σε περισσότερα datasets καθώς το διαθέσιμο εύρος αυξανόταν. Ωστόσο, η παρουσία θορύβου άλλαξε δραματικά αυτή τη δυναμική: στα δεδομένα με θόρυβο 30%, οι στατικές μέθοδοι best k για Conventional και Symmetric k-NN αναδείχθηκαν ως οι πιο αποτελεσματικές (μέσες κατατάξεις 4.50 και 4.61), υποδηλώνοντας ότι η παρουσία θορύβου ευνοεί την προσεγγισμένη επιλογή βέλτιστου k ανά dataset έναντι των πιο πολύπλοκων adaptive στρατηγικών. Το Wilcoxon signed-rank test επιβεβαίωσε αυτά τα ευρήματα, αποκαλύπτοντας ότι στα καθαρά δεδομένα οι στατιστικά σημαντικές διαφορές εντοπίζονται κυρίως στις συγκρίσεις Conventional vs Symmetric για τις adaptive μεθόδους, ενώ με την εισαγωγή θορύβου παρατηρήθηκε αύξηση των στατιστικά σημαντικών διαφορών, ιδίως για τη σύγκριση Conventional vs Mutual. Η χρήση του $k_{max} = \sqrt{n/2}$ επιβεβαίωσε τη γενική τάση αλλά με δραματική αύξηση του υπολογιστικού κόστους, με τον Mutual k-NN να απαιτεί έως 49 φορές περισσότερο χρόνο στα μεγάλα datasets. Συνολικά, τα αποτελέσματα υποδεικνύουν ότι οι adaptive μέθοδοι προσφέρουν σημαντικά οφέλη σε καθαρά δεδομένα με πολύπλοκη δομή, αλλά η αποτελεσματικότητά τους μειώνεται παρουσία θορύβου, όπου οι απλούστερες μέθοδοι με σωστή επιλογή του k τείνουν να είναι πιο σταθερές και αξιόπιστες.

Αναφορικά με τα κίνητρα και τη συνεισφορά της εργασίας, επιτεύχθηκαν οι αρχικοί στόχοι που τέθηκαν. Η ανάγκη για εις βάθος κατανόηση των αλγορίθμων k-NN και των σύγχρονων παραλλαγών τους, καθώς και η επιθυμία συστηματικής αξιολόγησης της απόδοσής τους σε πραγματικά προβλήματα ταξινόμησης, ικανοποιήθηκε μέσω της ολοκληρωμένης πειραματικής μελέτης σε 14 datasets υπό διαφορετικές συνθήκες. Η εργασία αντιμετώπισε τα κύρια ζητήματα που αναγνωρίστηκαν ως περιορισμοί του κλασικού k-NN—την ευαισθησία στον θόρυβο, τη σταθερότητα του k για όλα τα δείγματα, και την απουσία φιλτραρίσματος γειτονικών σχέσεων—μέσω της συστηματικής σύγκρισης των Mutual, Symmetric και adaptive προσεγγίσεων. Η κύρια συνεισφορά της εργασίας εντοπίζεται στην ανάπτυξη και υλοποίηση των δύο νέων υβριδικών αλγορίθμων Ada Mutual k-NN και Ada Symmetric k-NN, οι οποίοι αποτελούν πρωτότυπες προτάσεις που συνδυάζουν την προσαρμοστικότητα του AdaNN με τις έννοιες της αμοιβαιότητας και της συμμετρίας. Παρά το γεγονός ότι οι adaptive μέθοδοι δεν υπερτέρησαν καθολικά σε όλα τα σενάρια, ιδιαίτερα παρουσία θορύβου, η εργασία επέκτεινε τη βιβλιογραφία των αλγορίθμων k-NN και παρέχει νέες προοπτικές για την κατανόηση του πώς διαφορετικές στρατηγικές επιλογής γειτόνων και δυναμικής προσαρμογής του k συμπεριφέρονται υπό διαφορετικές συνθήκες δεδομένων. Τα ευρήματα υποδεικνύουν ότι δεν υπάρχει μία καθολικά βέλτιστη μέθοδος, αλλά η επιλογή εξαρτάται από τα χαρακτηριστικά των δεδομένων, την παρουσία θορύβου, και την ισορροπία μεταξύ ακρίβειας και υπολογιστικού κόστους που απαιτείται για κάθε εφαρμογή.

7.2 Προτάσεις για Μελλοντική Έρευνα

Η παρούσα εργασία αποτελεί μια πρώτη συστηματική προσπάθεια συνδυασμού της ιδέας του δυναμικού k με τις έννοιες της αμοιβαιότητας (Mutual k-NN) και της συμμετρίας (Symmetric k-NN), δημιουργώντας τους υβριδικούς αλγορίθμους Ada Mutual k-NN και Ada Symmetric k-NN. Τα αποτελέσματα των πειραμάτων αποκαλύπτουν ενδιαφέροντα μοτίβα απόδοσης και υποδεικνύουν ότι υπάρχει σημαντικό περιθώριο για περαιτέρω έρευνα και βελτιστοποίηση. Οι προτεινόμενες μέθοδοι επιδεικνύουν διαφορετική συμπεριφορά ανάλογα με τα χαρακτηριστικά των δεδομένων και την παρουσία θορύβου, γεγονός που ανοίγει νέους δρόμους για την ανάπτυξη πιο εξελιγμένων παραλλαγών που θα μπορούσαν να προσαρμόζονται ακόμη καλύτερα στην τοπική δομή των δεδομένων. Μελλοντική έρευνα θα μπορούσε να επικεντρωθεί στην ανάπτυξη νέων υβριδικών μεθόδων που συνδυάζουν επιπλέον τεχνικές φιλτραρίσματος γειτόνων ή εναλλακτικά κριτήρια για την επιλογή του προσαρμοστικού k, καθώς και στη θεωρητική ανάλυση των προτεινόμενων προσεγγίσεων για την καθιέρωση φραγμάτων σφάλματος και συνθηκών βέλτιστης απόδοσης. Η βελτιστοποίηση της υπολογιστικής αποδοτικότητας αποτελεί επίσης κρίσιμη κατεύθυνση, ιδιαίτερα για τις adaptive μεθόδους που παρουσιάζουν σημαντικό υπολογιστικό κόστος, ώστε να καταστούν πρακτικές για εφαρμογές πραγματικού χρόνου ή για datasets πολύ μεγάλης κλίμακας.

Βιβλιογραφία

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, jan 1967.

[2] M. Papanikolaou, G. Evangelidis, and S. Ougiaroglou, "Dynamic k determination in k-nn classifier: A literature review," *2021 IEEE International Conference on Technology and Entrepreneurship (ICTE)*, pp. 1–8, 2021.

[3] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 1, pp. 91–94, IEEE, 2010.

[4] Z. Pan, Y. Wang, and Y. Pan, "A new locally adaptive k-nn algorithm based on discrimination class," *Knowledge-Based Systems*, vol. 204, p. 106185, 2020.

[5] O. Anava and K. Y. Levy, "k*-nearest neighbors: From global to local," in *Advances in Neural Information Processing Systems*, vol. 29, pp. 4923–4931, 2016.

[6] F. Di Salvo, S. Doerrich, I. Rieger, and C. Ledig, "An embedding is worth a thousand noisy labels," *arXiv preprint arXiv:2408.14358*, 2024.

[7] S. S. Mullick, S. Datta, and S. Das, "Adaptive learning-based k-nearest neighbor classifiers with resilience to class imbalance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5713–5725, 2018.

[8] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Learning k for knn classification," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 3, pp. 1–19, 2017.

[9] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Efficient knn classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018.

[10] N. Garcia-Pedrajas, J. A. Romero del Castillo, and G. Cerruela-Garcia, "A proposal for local k values for k-nearest neighbor rule," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 470–475, 2017.

[11] F. Bulut and M. F. Amasyali, "Locally adaptive k parameter selection for nearest neighbor classifier: one nearest cluster," *Pattern Analysis and Applications*, vol. 20, no. 2, pp. 415–425, 2017.

[12] S. Ougiaroglou, G. Evangelidis, and K. I. Diamantaras, "Dynamic k-nn classification based on region homogeneity," in *New Trends in Databases and Information Systems*, pp. 27–37, Springer, 2020.

[13] A. K. Ghosh, "On optimum choice of k in nearest neighbor classification," *Computational Statistics & Data Analysis*, vol. 50, no. 11, pp. 3113–3123, 2006.

[14] A. K. Ghosh, "On nearest neighbor classification using adaptive choice of k," *Journal of Computational and Graphical Statistics*, vol. 16, no. 2, pp. 482–502, 2007.

[15] U. Johansson, H. Boström, and R. König, "Extending nearest neighbor classification with spheres of confidence," in *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, pp. 282–287, 2008.

[16] G. Bhattacharya, K. Ghosh, and A. S. Chowdhury, "Test point specific k estimation for knn classifier," in *2014 22nd International Conference on Pattern Recognition*, pp. 1478–1483, IEEE, 2014.

[17] G. Bhattacharya, K. Ghosh, and A. S. Chowdhury, "A probabilistic framework for dynamic k estimation in knn classifiers with certainty factor," in *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pp. 1–5, IEEE, 2015.

[18] T. I. Cannings, T. B. Berrett, and R. J. Samworth, "Local nearest neighbour classification with applications to semi-supervised learning," *The Annals of Statistics*, vol. 48, no. 3, pp. 1789–1814, 2020.

[19] A. Balsubramani, S. Dasgupta, and S. Moran, "An adaptive nearest neighbor rule for classification," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] H. Döring, L. Györfi, and H. Walk, "Rate of convergence of k-nearest-neighbor classification rule," *Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2651–2695, 2018.

[21] P. Zhao and L. Lai, "Efficient classification with adaptive knn," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11007–11014, 2021.

[22] S. Uddin, I. Haque, H. Lu, M. A. Moni, and E. Gide, "Comparative performance analysis of k-nearest neighbour (knn) algorithm and its different variants for disease prediction," *Scientific Reports*, vol. 12, no. 1, p. 6256, 2022.

[23] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pp. 986–996, Springer, 2003.

[24] J. Wang, P. Neskovic, and L. N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure," *Pattern Recognition Letters*, vol. 28, no. 2, pp. 207–213, 2007.

[25] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "Iknn: Informative k-nearest neighbor pattern classification," in *Knowledge Discovery in Databases: PKDD 2007*, pp. 248–264, Springer, 2007.

[26] J. Li, Y. Shyr, and Q. Liu, "Single-cell and spatial transcriptomics clustering with an optimized adaptive k-nearest neighbor graph," *bioRxiv*, 2023.

[27] L. Baoli, L. Qin, and Y. Shiwen, "An adaptive k-nearest neighbor text categorization strategy," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 4, pp. 215–226, 2004.

[28] H. Liu, S. Zhang, J. Zhao, X. Zhao, and Y. Mo, "A new classification algorithm using mutual nearest neighbors," in *Proceedings of the 9th International Conference on Grid and Cloud Computing*, (Nanjing, China), pp. 52–57, IEEE, 2010.

[29] R. Nock, M. Sebban, and P. Jappy, "A symmetric nearest neighbor learning rule," in *Advances in Case-Based Reasoning: 5th European Workshop, EWCBR 2000*, vol. 1898 of *Lecture Notes in Computer Science*, (Trento, Italy), pp. 241–252, Springer, 2000.

[30] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[31] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," 2009.

[32] R. K. Bock, "Magic gamma telescope dataset." UCI Machine Learning Repository, 2007.

[33] F. Alimoglu and E. Alpaydin, "Pen-based recognition of handwritten digits dataset." UCI Machine Learning Repository, 1998.

[34] "Statlog (landsat satellite) dataset." UCI Machine Learning Repository, 1993.

[35] E. Project, "Phoneme dataset." KEEL Repository, 1994.

[36] G. Rätsch, T. Onoda, and K. R. Müller, "Banana dataset." KEEL Repository, 2001.

[37] P. Horton and K. Nakai, "Yeast dataset." UCI Machine Learning Repository, 1996.

[38] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Pima indians diabetes dataset." UCI Machine Learning Repository, 1988.

[39] "Texture dataset." KEEL Repository, 1999.

[40] "Statlog (shuttle) dataset." UCI Machine Learning Repository, 1993.

[41] P. Horton and K. Nakai, "E. coli dataset." UCI Machine Learning Repository, 1996.

[42] R. A. Fisher, "Iris dataset." UCI Machine Learning Repository, 1936.

[43] L. Breiman, “Twonorm dataset.” KEEL Repository, 1998.

[44] R. S. Siegler, “Balance scale dataset.” UCI Machine Learning Repository, 1994.

[45] P. W. Frey and D. J. Slate, “Letter recognition dataset.” UCI Machine Learning Repository, 1991.