Διεθνές Πανεπιστήμιο
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΊΑ

Προσδιορισμός της παραμέτρου k στον κατηγοριοποιητή κ εγγύτερων γειτόνων μέσω της τεχνικής AdaNN και διερεύνηση υλοποίησης παραλλαγών της

Φοιτητής:
Καρδαμανίδης Χρήστος
ΑΜ: 062021

Επιβλέπων:
Στέφανος Ουγιάρογλου

20 Σεπτεμβρίου 2023

Title of Dissertation Specifying the parameter k value in k-NN classification through AdaNN and implementation of new variants
Code of Dissertation 22280
Student's full name Kardamanidis Christos
Supervisor's full name Ougiaroglou Stefanos
Date of undertaking 21-10-2022
Date of completion 20-09-2023

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

Η παρούσα διπλωματική εργασία με θέμα «Προσδιορισμός της παραμέτρου k στον κατηγοριοποιητή κ εγγύτερων γειτόνων μέσω της τεχνικής AdaNN και διερεύνηση υλοποίησης παραλλαγών της» πραγματοποιήθηκε για το ΠΜΣ Ευφυείς Τεχνολογίες Διαδικτύου του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστήμιου Ελλάδος. Θα ήθελα να δώσω ιδιαίτερες ευχαριστίες μου στον καθηγητή κύριο Ουγιάρογλου Στέφανο που μου έδωσε την ευκαιρία υπό την καθοδήγησή του να ολοκληρώσω αυτήν την διπλωματική εργασία. Η σημαντική βοήθεια και οι πολύτιμες συμβουλές του μου ήταν ιδιαίτερα χρήσιμες τόσο για την ολοκλήρωση της διπλωματικής μου αλλά και για την μετέπειτα εξέλιξη μου στην επιστήμη της πληροφορικής. Θα ήθελα επίσης να ευχαριστήσω όλους τους καθηγητές και το προσωπικό του τμήματος αλλά και τους συμφοιτητές μου για τις πολύτιμες γνώσεις που μου προσέφεραν σε όλη αυτήν την χρονική διάρκεια του μεταπτυχιακού προγράμματος. Τέλος, το πιο μεγάλο ευχαριστώ είναι για τα δικά μου πρόσωπα και στους γονείς μου για την καθοδήγηση και την στήριξή τους σε όλη αυτήν την χρονική περίοδο του μεταπτυχιακού μου προγράμματος.

Περίληψη

Στον τομέα της επιστήμης των δεδομένων μια πολύ γνωστή παραλλαγή του αλγορίθμου KNN είναι ο αλγόριθμος AdaNN. Ο αλγόριθμος KNN εδώ και πολλά χρόνια είναι ένας ευρέως χρησιμοποιούμενος αλγόριθμος ο οποίος έχει κάποια πλεονεκτήματα και μειονεκτήματα. Ένα από τα βασικά του μειονεκτήματα είναι ο σταθερός αριθμός του κ, ο οποίος καθορίζει το μέγεθος της κάθε γειτονιάς στιγμιότυπου και συνεπώς επηρεάζει το τελικό αποτέλεσμα της ταξινόμησης του στιγμιότυπου. Μια βασική παραλλαγή του αλγορίθμου είναι ο AdaNN ο οποίος ορίζει διαφορετικό κ για κάθε στιγμιότυπο του dataset. Σκοπός της διπλωματικής είναι να γίνει μια υλοποίηση του βασικού αλγορίθμου AdaNN και να γίνει σύγκριση της απόδοσής του αλγορίθμου AdaNN με τον KNN. Επίσης στην παρούσα διπλωματική θα γίνει μια διερεύνηση και υλοποίηση νέων παραλλαγών του αλγορίθμου AdaNN στην προγραμματιστική γλώσσα Python, θα αναλυθεί η μεθοδολογία της κάθε παραλλαγής και θα γίνει σύγκριση με τα αποτελέσματά της κάθε εκδοχής με την απλή εκδοχής του αλγορίθμου AdaNN καθώς και σύγκριση με τον KNN.

Λέξεις Κλειδιά: AdaNN, Δυναμικό κ, KNN, Κατηγοριοποίηση, Εξώρυξη Δεδομένων

Abstract

In the field of data science a well-known variant of the KNN algorithm is the AdaNN algorithm. The KNN algorithm is a widely used algorithm for many years, which has some advantages and disadvantages. One of its main disadvantages is the fixed number of k, which determines the size of each instance's neighborhood and therefore affects the final result of instance classification. A basic variant of this algorithm is AdaNN, which sets a different k for each instance of the dataset. The purpose of the thesis is to make an implementation of the basic AdaNN algorithm and to compare the performance of each variant with the AdaNN algorithm and the traditional KNN. Also in this thesis, an investigation and implementation of new variants of the AdaNN algorithm will be implemented in the Python programming language, the methodology of each variant will be analyzed and a comparison will be made with the results of the simple version of the AdaNN algorithm as well as a comparison with KNN.
Keywords: AdaNN, Dynamic K, KNN, Classification, Data Mining

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

Στην πρώτη αυτή ενότητα της εισαγωγής θα γίνει μια αρχική προσέγγιση στην διπλωματική προσπαθώντας να παρουσιαστεί το πρόβλημα το οποίο πάει να επιλύσει η διπλωματική, καθώς επίσης θα δοθεί ένα πλάνο για την ανάλυση της επίλυσης αυτού του προβλήματος. Είναι σημαντικό να αναφερθεί το κίνητρο για την επίλυση αυτού του προβλήματος αλλά και το ποια θα είναι τελικά η συνεισφορά της διπλωματικής στην γνώση της επιστήμης των δεδομένων. Επίσης στην φάση της εισαγωγής θα γίνει ανάλυση σε θεωρητικό επίπεδο σε θέματα τα οποία είναι χρήσιμα για την περαιτέρω κατανόηση της διπλωματικής. Τέτοια θέματα θα αφορούν την κατηγοριοποίηση η οποία είναι θέμα στο οποίο θα επικεντρωθεί η διπλωματική και συνεπώς θα πρέπει να γίνει ιδιαίτερη ανάλυση σε αυτήν, καθώς επίσης και στα είδη των κατηγοριοποιητών τα οποία υπάρχουν.

1.2 Κατηγοριοποίηση

Το πρώτο σημείο στο οποίο υπάρχει ανάγκη για εκτενή αναφορά και ανάλυση είναι η κατηγοριοποίηση. Η ταξινόμηση είναι μια κατηγορία προβλήματος η οποία πάει να επιλυθεί στον ευρύτερο τομέα της επιστήμης των δεδομένων αλλά και της μηχανικής μάθησης. Η διαδικασία η οποία υπάρχει για την ταξινόμηση αυτή λέγεται κατηγοριοποίηση. Η διαδικασία αυτή έχει ως στόχο να δημιουργήσει μια απεικόνιση ενός συνόλου δεδομένων σε κάποιες ήδη προκαθορισμένες ομάδες. Αυτές οι ομάδες μπορούν συχνά στην βιβλιογραφία να αναφέρονται ως κλάσεις ή κατηγορίες.
Ιδιαίτερα στην επιστήμη της εξόρυξης γνώσης από τα δεδομένα αλλά και στην μηχανική μάθηση για να προχωρήσουμε στην φάση της κατηγοριοποίησης προηγείται μια άλλη διαδικασία, η φάση της εκπαίδευσης. Γενικά υπάρχουν διάφοροι τρόποι εκπαίδευσης όπως η μάθηση με επίβλεψη, η μάθηση χωρίς επίβλεψη και η ενισχυτική μάθηση. Ιδιαίτερη βαρύτητα θα δοθεί στην κατηγοριοποίηση με επίβλεψη στην οποία επικεντρώνεται ο αλγόριθμος Κ εγγύτερων γειτόνων (K-NN) αλλά και ο AdaNN καθώς και οι παραλλαγές οι οποίες θα παρουσιαστούν στην παρούσα διπλωματική. Ο τρόπος που μαθαίνει ο αλγόριθμος KNN αλλά και ο AdaNN καθώς και οι παραλλαγές μας είναι με επίβλεψη. Η μάθηση αυτή έχει 2 σκοπούς. Αυτοί οι 2 σκοποί έχουν να κάνουν με τις 2 φάσεις τις οποίες περιλαμβάνει η μάθηση με επίβλεψη, όπου η πρώτη

είναι το χτίσιμο ενός μοντέλου το οποίο στην συνέχεια θα χρησιμοποιηθεί για την εξαγωγή συμπερασμάτων. Κατά την πρώτη φάση θα πρέπει να υπάρχει ένα σύνολο δεδομένων το οποίο να περιλαμβάνει παραδείγματα τα οποία είναι αντιπροσωπευτικά για κάθε κατηγορία που υπάρχει στο σύνολο των δεδομένων μας. Σημαντικό να αναφερθεί είναι πως κάθε οντότητα του συνόλου δεδομένων που χρησιμοποιούμε είναι επίσης γνωστό ως στιγμιότυπο (instance). Κατά την δεύτερη φάση της μάθησης με επίβλεψη και εφόσον έχει φτιαχτεί το μοντέλο κατηγοριοποίησης, αυτό το χρησιμοποιούμε για την εξαγωγή συμπεράσματος για ένα καινούργιο στιγμιότυπο για το οποίο δεν γνωρίζουμε την κατηγορία του. Με αυτόν τον τρόπο ο αλγόριθμος μπορεί με βάση τα ήδη υπάρχοντα παραδείγματα που του έχουμε δώσει στην εκπαίδευση να αντιληφθεί σε ποια κλάση θα ανήκει αυτό το καινούργιο δεδομένο. Η διαφορά των δυο αυτών φάσεων είναι πως στην φάση του χτισίματος του μοντέλου έχουμε μια επαγωγική διαδικασία ενώ στην φάση της κατηγοριοποίησης έχουμε μια συμπερασματική διαδικασία. [1]
Εκτός από τις φάσεις της εκπαίδευσης και τη εξαγωγής συμπεράσματος, μπορούμε να διακρίνουμε αρκετές φάσεις οι οποίες χρειάζονται προκειμένου να φτιαχτεί ένα καλό μοντέλο. Το πρώτο στάδιο είναι η συλλογή των δεδομένων, το οποίο περιγράφει την συγκέντρωση του συνόλου δεδομένων. Το σύνολο των δεδομένων θα πρέπει να περιλαμβάνει επαρκή παραδείγματα για κάθε κλάση που θα υπάρχει στο σύστημα μας. Θα πρέπει το dataset με το οποίο θα φτιαχτεί το σύστημα να είναι ισορροπημένο, δηλαδή να μην περιλαμβάνουν πιο πολλά στιγμιότυπα από την μια κλάση και λίγα από την άλλη, θα πρέπει δηλαδή να υπάρχει μια ισορροπία στους αριθμούς των στιγμιοτύπων των κλάσεων. Επίσης, ανάλογα με τον τύπο του προβλήματος θα πρέπει να λαμβάνονται υπόψη χαρακτηριστικά για την δειγματοληψία των δεδομένων. Για παράδειγμα, δεν έχει νόημα να γίνει πρόβλεψη για κατηγοριοποίηση ενός λουλουδιού σε ένα οικοσύστημα και το ίδιο σύστημα να μπορέσει να λειτουργήσει σωστά σε ένα άλλο οικοσύστημα. Το αμέσως επόμενο σημαντικό στοιχείο μετά την συλλογή των δεδομένων είναι η προεπεξεργασία τους. Αυτή η διαδικασία περιλαμβάνει τον καθαρισμό των δεδομένων, δηλαδή να χειριστούμε τυχόν τιμές οι οποίες λείπουν ή ακόμα και να γίνει μετατροπή των δεδομένων τα οποία να είναι σε μορφή κατάλληλη για ανάλυση. Στην φάση της διαχείρισης ελλειπόντων τιμών είναι δυνατό να γίνει μια γκάμα ενεργειών όπως η διαγραφή των εγγραφών που έχουν ελλειπή δεδομένα ή ακόμα να δημιουργηθεί μια εκτιμώμενη τιμή για να συμπληρωθούν αυτές οι τιμές. Όσον αφορά την φάση του μετασχηματισμού των δεδομένων υπάρχει μια πληθώρα ενεργειών όπως η κανονικοποίηση και η διακριτοποίηση των τιμών. Ένα τέτοιο παράδειγμα είναι η μετατροπή μιας συνεχούς τιμής σε κάποιες συγκεκριμένες τιμές, όπως για παράδειγμα η διακριτοποίηση του εισοδήματος μιας οικογένειας σε φορολογικές κλίμακες με συγκεκριμένους αριθμούς. Ένα παράδειγμα κανονικοποίησης των αριθμητικών τιμών είναι με την κανονική κατανομή όπου θα μπορούσε να γίνει μετατροπή μιας αριθμητικής τιμής στο διάστημα [0-1]. Το αμέσως επόμενο στάδιο πριν την κατηγοριοποίηση είναι η επιλογή χαρακτηριστικών τα οποία είναι σχετικά για την εξαγωγή του αποτελέσματος. Ιδιαίτερα σημαντικό είναι να αναφερθεί πως η διπλωματική δεν θα δώσει ιδιαίτερη βάση στην συλλογή, προ-επεξεργασία και επιλογή χαρακτηριστικών αλλά κυρίως θα κινηθεί στις παραμέτρους επιλογής αλγορίθμου, της εκπαίδευσης και της αξιολόγησης [2].

- **Συλλογή Δεδομένων**
- **Προ-επεξεργασία Δεδομένων**
- **Επιλογή Χαρακτηριστικών**

- **Επιλογή αλγορίθμου**
- **Εκπαίδευση**
- **Εκτίμηση Δεδομένων**
- **Έλεγχος του Μοντέλου**

Ιδιαίτερη αναφορά πρέπει να γίνει στα είδη προβλημάτων κατηγοριοποίησης τα οποία μπορούν να αντιμετωπιστούν από έναν κατηγοριοποιητή. Το πρόβλημα της κατηγοριοποίησης έχει να κάνει με τον αριθμό των κλάσεων αλλά και με την πολλαπλότητα των κλάσεων ανά στιγμιότυπο. Η πιο απλή περίπτωση είναι τα δυαδικά προβλήματα κατηγοριοποίησης (binary classification) στην οποία στα δεδομένα μας υπάρχουν 2 κλάσεις και κάθε στιγμιότυπο το οποίο υπάρχει στο σύνολο δεδομένων μας θα ανήκει σε μια από αυτές. Σε μια άλλη περίπτωση θα μπορούσαν να υπάρχουν παραπάνω από 2 κατηγορίες και κάθε ένα στιγμιότυπο θα ανήκει σε κάποια από αυτές. Τέλος σε μια πιο σύνθετη περίπτωση θα μπορούσαν να υπάρχουν παραπάνω από δυο κλάσεις και ένα στιγμιότυπο να ανήκει σε περισσότερες από μια κλάσεις.

1.3 Τύποι κατηγοριοποιητών

Στην προηγούμενη ενότητα γίνεται αναφορά στην έννοια του κατηγοριοποιητή. Ο κατηγοριοποιητής όπως γίνεται αντιληπτό είναι το σύστημα το οποίο θα κάνει κατηγοριοποίηση ένα στιγμιότυπο. Στην επιστήμη της εξόρυξης των δεδομένων αλλά και της μηχανικής μάθησης όταν μιλάμε για κατηγοριοποιητές αναφερόμαστε συνήθως σε είδη κατηγοριοποιητών τους οκνηρούς (lazy) και τους πρόθυμους (eager). Και οι δυο κατηγοριοποιητές έχουν ίδιο σκοπό, ο οποίος είναι να γίνει πρόβλεψη της κλάσης του στιγμιότυπου αλλά έχουν διαφορά όσον αφορά τις λειτουργίες τους. Ο πρόθυμος κατηγοριοποιητής με βάση τα υπάρχοντα δεδομένα εκπαίδευσης φτιάχνει ένα μοντέλο το οποίο θα κατηγοριοποιεί κάθε νέο στιγμιότυπο. Χαρακτηριστικά παραδείγματα πρόθυμο κατηγοριοποιητή είναι το MLP και το Naive Bayes. Σε αντίθεση με τους πρόθυμους κατηγοριοποιητές ο KNN ο οποίος αποτελεί την βάση του AdaNN αλλά και των παραλλαγών της διπλωματικής ανήκει στους οκνηρούς κατηγοριοιποιητές, δηλαδή σε αυτούς οι οποίοι δεν φτιάχνουν ένα μοντέλο. Αυτό το οποίο θεωρούν ως μοντέλο είναι τα δεδομένα τα οποία χρησιμοποιήθηκαν κατά την φάση της εκπαίδευσης. Έτσι με αυτήν την λογική ένα αλγόριθμος ο οποίος είναι οκνηρός θα κατηγοριοποίησει ένα νέο στιγμιότυπο κατά την στιγμή που θα φτάσει με τον έλεγχο όλου του συνόλου εκπαίδευσης. [3]. Ουσιαστικά στους οκνηρούς αποφεύγεται η δημιουργία γενικών κανόνων ή μοντέλων τα οποία θα κάνουν κατηγοριοποίηση ενώ στους οκνηρούς κατά την στιγμή της εκπαίδευσης θα φτιαχτεί μοντέλο. Αυτή τους η διαφορά έχει άμεσες επιδράσεις στην απόδοση της μάθησης αλλά και την προσαρμογή του αλγορίθμου σε νέα δεδομένα καθώς και στο πόσο ευέλικτα είναι τα παραγόμενα μοντέλα. Όσον αφορά το ιστορικό κομμάτι οι πρόθυμοι κατηγοριοποιητές έκαναν την εμφάνιση τους πριν από τους οκνηρούς, αλλά βρέθηκαν αντιμέτωποι με προκλήσεις σε περιπτώσεις που υπάρχουν μεγάλα σύνολα δεδομένων ή όταν υπάρχει ανάγκη για προβλέψεις κατηγοριοποίησης σε πραγματικό χρόνο.Η προσπάθεια επίλυσης τέτοιου είδους προβλημάτων οδήγησε στην δημιουργία των οκνηρών κατηγοριοποιητών.

Σε αυτό το σημείο είναι ιδιαίτερα σημαντικό να γίνει μια σύγκριση μεταξύ αυτών των ειδών κατηγοριοποιητών. Ανάλογα με το είδος του κατηγοριοποιητή μπορούμε να πούμε πως υπάρχει και διαφορετικό είδος μάθησης. Το πρώτο σημείο στο οποίο πρέπει να σταθούμε είναι η προσαρμοστικότητα του αλγορίθμου. Προσαρμοστικότητα θεωρούμε την ικανότητα ενός αλγορίθμου να εφαρμόζεται γρήγορα σε καινούρια δεδομένα. Σε αυτό το κριτήριο οι οκνηροί κατηγοριοποιητές έχουν πλεονέκτημα καθώς το ποια αντικείμενα θα χρειαστεί καθορίζεται την ώρα που θα γίνει η κλήση για εξαγωγή συμπεράσματος. Αυτό είναι χρήσιμο καθώς μπορεί απλά να γίνει αποθήκευση μόνο των νέων δεδομένων πράγμα το οποίο δεν συμβαίνει στους πρόθυμους κατηγοριοποιητές καθώς μετά από την ενσωμάτωσή νέων δεδομένων το σύστημα χρειάζεται καινούργια εκπαίδευση. Ένα άλλο σημείο στο οποίο επίσης έχουν πλεονέκτημα οι οκνηροί κατηγοριοποιητές είναι ότι φτιάχνουν πιο απλό μοντέλο. Αυτό συμβαίνει καθώς σε έναν οκνηρό κατηγοριοποιητή γίνεται έλεγχος μόνο της γειτονιάς του στιγμιότυπου προς εξέταση, σε αντίθεση με τον πρόθυμο κατηγοριοποιητή ο οποίος προσπαθεί να βρει πιο γενικά μοτίβα σε όλο το σύνολο δεδομένων. Η απλότητα ενός μοντέλου έχει άμεση σχέση και με τον χρόνο της εκπαίδευση του αλγορίθμου καθώς ο οκνηρός επικεντρώνεται στην αποθήκευση των δεδομένων τα οποία να μπορούν να γίνουν ανάκτηση πιο μετά ενώ ο πρόθυμος κατηγοριοποιητής έχει μια ολόκληρη φάση που θα κάνει εκπαίδευση. Επίσης άλλο σημαντικό πλεονέκτημα είναι ότι οι οκνηροί κατηγοριοποιητές είναι κατάλληλοι για πρόβλεψη σε πραγματικό χρόνο καθώς μπορεί γρήγορα να αποθηκεύσει νέα δεδομένα ενώ οι πρόθυμοι κατηγοριοποιητές θα χρειαστούν χρόνο για την επανεκπαίδευση του μοντέλου τους. Σε αυτό το σημείο είναι αναγκαίο να αναφερθούν και τα μειονεκτήματα των οκνηρών κατηγοριοποιητών. Τέτοια παραδείγματα είναι η χρήση της μνήμης καθώς και η ευαισθησία που έχουν στον θόρυβο, Επίσης όσον αφορά τον χρόνο που γενικεύει ο αλγόριθμος οι οκνηροί κατηγοριοποιητές φαίνεται να έχουν μειονέκτημα σε σχέση με τους πρόθυμους καθώς κάνουν αναζήτηση μέσα σε όλα τα δεδομένα για να βρουν τα άλλα σχετικά στιγμιότυπα της γειτονιάς που ανήκουν κάτι το οποίο δεν συμβαίνει στους πρόθυμους κατηγοριοποιητές καθώς έχουν ένα έτοιμο προ-εκπαιδευμένο μοντέλο. Άρα λαμβάνοντας υπόψη όλα τα παραπάνω φαίνεται ότι η πρόθυμη μάθηση (eager learning) μπορεί να πετύχει πολύ καλές προβλέψεις, οι οποίες όμως έχουν άμεση εξάρτηση από το μοντέλο και την εκπαίδευση του πράγμα το οποίο δεν συμβαίνει στην οκνηρή μάθηση. Από την άλλη μεριά η οκνηρή μάθηση έχει πολύ μεγαλύτερη προσαρμοστικότητα, μπορώντας να χειριστεί μια ποικιλία από διαφορετικά είδη δεδομένων. Κάποια χαρακτηριστικά παραδείγματα οκνηρών κατηγοριοποιητών είναι ο K-NN, ο CBR, το LVQ και τα δίκτυα RBF. Ο πρώτος αλγόριθμος είναι ο K - εγγύτερων γειτόνων ο οποίος κάνει αναζήτηση στο σύνολο των δεδομένων εκπαίδευσης και χρησιμοποιώντας μια μετρική εγγύτητά θα επιλέξει την κατηγορία στην οποία ανήκουν οι κ κοντινότεροι του γείτονες. Από την άλλη ο Case-Based Reasoning κάνει ανάκτηση και επαναχρησιμοποιεί ήδη υπάρχοντα στιγμιότυπα έχοντας ως σκοπό την επίλυση νέων προβλημάτων. Το Learning Vector Quantization (LVQ) χρησιμοποιεί κάποια διανύσματα κατά την εκπαίδευση ώστε να απεικονίσει τις διάφορες κατηγορίες. Στην συνέχεια αυτό που συμβαίνει είναι ότι όταν έρχεται ένα στιγμιότυπο το κατηγοριοποιεί με βάση το κοντινότερο διάνυσμα. Τέλος είναι τα δίκτυα Radial Basis FUnction με τα οποία αναφερόμαστε σε μια κατηγορία νευρωνικών δικτύων τα οποία χρησιμοποιούν συναρτήσεις ακτινικής βάσης [4].
Σε αυτό το σημείο είναι σημαντικό εφόσον έχει γίνει η σύγκριση των δυο ειδών κατηγοριοποιητών να γίνει μια αναφορά και στον τρόπο με τον οποίο θα γίνει η επιλογή του είδους αυτού. Αρχικά στους οκνηρούς κατηγοριοποιητές βλέπουμε πως είναι πιο κατάλληλοι για εφαρμογές που έχουν να κάνουν με πιο πολλά δεδομένα. Δηλαδή, ένας κατηγοριοποιητής ο οποίος είναι

Χαρακτηριστικό	Οκνηροί Κατηγοριοποιητές	Πρόθυμοι Κατηγοριοποιητές
Προσαρμοστικότητα	Μεγαλύτερη	Μικρότερη
Πολυπλοκότητα Μοντέλου	Μικρότερη	Μεγαλύτερη
Χρόνος Εκπαίδευσης	Μικρότερος	Μεγαλύτερος
Χρόνος Πρόβλεψης	Μεγαλύτερος	Μικρότερος
Χρήση Μνήμης	Μεγαλύτερη	Μικρότερη
Ερμηνεία Μοντέλου	Μεγαλύτερη	Μικρότερη
Στιβαρότητα Αλγορίθμου	Μικρότερη	Μεγαλύτερη

Πίνακας 1.1: Σύγκριση Οκνηρών και Πρόθυμων Κατηγοριοποιητών.

οκνηρός είναι πιο κατάλληλος για χρήση συνόλων δεδομένων μεγαλύτερης κλίμακας, όπως για παράδειγμα την ανάλυση των κοινωνικών δικτύων αλλά και διάφορα δίκτυα αισθητήρων. Αυτά έχουν το χαρακτηριστικό ότι επειδή δέχονται πάρα πολλά δεδομένα σε πραγματικό χρόνο είναι κατάλληλα για να κάνουν τροποποίηση στα ήδη υπάρχοντα δεδομένα. Σε ένα τέτοιο παράδειγμα εφαρμογής ένα μοντέλο πρόθυμου κατηγοριοποιητή δεν θα ήταν κατάλληλο καθώς υπάρχει η απαίτηση επανεκπαίδευσης ανά διαστήματα. Από την άλλη πλευρά ένας κατηγοριοποιητής οποίος χτίζει μοντέλο, δηλαδή κάνει πρόθυμη μάθηση θα αργεί περισσότερο κατά την φάση της εκπαίδευσης αλλά θα έχει καλύτερους χρόνους κατηγοριοποίησης καθώς θα μπορεί να κάνει προβλέψεις σε πραγματικό χρόνο. Παραδείγματα τέτοιων συστημάτων που χρειάζονται γρήγορη απόκριση είναι η αναγνώριση ενός αμαξιού το οποίο ανιχνεύει ένας αισθητήρας. Σε μια τέτοια περίπτωση είναι κρίσιμη η απόκριση σε πολύ μικρό χρονικό διάστημα.

1.4 Κίνητρο και Συνεισφορά

Ιδιαίτερα σημαντικό για την παρούσα διπλωματική είναι να γίνει κατανοητό το κίνητρο για την υλοποίηση της. Το βασικό κίνητρο το οποίο υπήρχε για την υλοποίηση της διπλωματικής αυτής είναι αρχικά το πρόβλημα το οποίο έχει ο παραδοσιακός αλγόριθμος KNN το οποίο έγκειται στην σταθερότητα της τιμής του κ για κάθε στιγμιότυπου του dataset. Το βασικό πρόβλημα αυτής της τακτικής είναι πως το dataset δεν έχει μια συγκεκριμένη ομοιογένεια και πυκνότητά και συνεπώς κάθε στιγμιότυπο ανάλογα με την γειτονιά που βρίσκεται είναι πιθανό να χρειάζεται διαφορετική τιμή του κ για την οποία θα μπορέσει να κατηγοριοποιηθεί σωστά. Για αυτόν τον λόγο, εφόσον υπάρχει ο αλγορίθμος AdaNN ο οποίος ορίζει μια διαφορετική τιμή του κ για κάθε στιγμιότυπο, θα πρέπει να υλοποιηθεί η κλασσική του εκδοχή και να παρουσιαστούν κάποιες παραλλαγές αυτού. Για αρχή είναι σημαντικό να αναφερθεί πως στο paper δεν γίνεται σύγκριση με το τον Best KNN [5]. Συνεπώς σε αυτή την διπλωματική θεωρούμε σημαντικό να γίνει μια σύγκριση του AdaNN με την καλύτερη δυνατή τιμή που έχει προκύψει από την διαδικασία του tuning. Επίσης ένα βασικό πρόβλημα της κλασσικής παραλλαγής του AdaNN είναι ότι για κάθε στιγμιότυπο όταν θα προσπαθήσει να κάνει διαδοχικές εκτελέσεις του αλγορίθμου KNN μέχρι την χρονική στιγμή που το στιγμιότυπο θα μπορεί να κατηγοριοποιηθεί σωστά φτάνει μέχρι μια προκαθορισμένη τιμή 9. Συνεπώς κάθε στιγμιότυπο μπορεί να φτάσει μέχρι την μέγιστη τιμή 9 την οποία μπορεί να λάβει προκειμένου με αυτήν να εκτελεστεί ο αλγόριθμος των κ εγγύτερων γειτόνων. Αυτό όμως είναι μια προκαθορισμένη τιμή η οποία πιθανό να μην

εφαρμόζεται σε διάφορα dataset τα οποία να έχουν πολύ μεγαλύτερες γειτονιές ή σε dataset τα οποία μπορεί να έχουν πολύ περισσότερα στιγμιότυπα. Για αυτόν τον λόγο, κατά την διάρκεια της διπλωματικής θα γίνει διερεύνηση αυτού του μέγιστου ορίου του μέχρι το οποίο μπορεί να φτάσει η τιμή του κ για ένα στιγμιότυπο. Κατά την διάρκεια αυτής της διπλωματικής θα γίνει διερεύνηση αυτού του μέγιστου ορίου με διάφορες συναρτήσεις υπολογισμού, με βάση κάποιους παράγοντες. Τέτοιοι παράγοντες είναι ο αριθμός των στιγμιοτύπων που έχει ένα dataset καθώς και πόσα χαρακτηριστικά έχει η κλάση στην οποία ανήκει το εξεταζόμενο στιγμιότυπο. Επίσης κάποιες άλλες παραλλαγές οι οποίες θα διερευνηθούν έχουν να κάνουν και με την αφαίρεση του θορύβου από το dataset. Όπως είναι λογικό υπάρχουν διάφορα στιγμιότυπα τα οποία μπορεί να αποτελούν θόρυβο. Σε τέτοιες περιπτώσεις όπως είναι λογικό, τέτοια στιγμιότυπα δεν θα μπορούν κατηγοριοποιηθούν μέχρι έναν φυσιολογικό αριθμό κ καθώς είναι πιθανό όλη του η γειτονιά να μην ανήκει στην κλάση στην οποία βρίσκεται. Για αυτόν τον λόγο θεωρήθηκε σημαντικό να γίνει διαχείριση τέτοιων στιγμιοτύπων με την αφαίρεση τους είτε με ένα είτε με πολλαπλά περάσματα.

1.5 Οργάνωση διπλωματικής εργασίας

Σε αυτό το σημείο εφόσον έχει γίνει μια εισαγωγή στο θέμα της διπλωματικής είναι ιδιαίτερα σημαντικό να γίνει μια ανάλυση της διάρθρωσης της διπλωματικής εργασίας. Το πρώτο κεφάλαιο το οποίο έχει ήδη αναφερθεί είναι η εισαγωγή στο θέμα της διπλωματικής εργασίας. Αυτό το κεφάλαιο περιλαμβάνει κάποιες εισαγωγικές έννοιες οι οποίες μας είναι απαραίτητες προκειμένου να γίνουν αντιληπτές κάποιες βασικές έννοιες οι οποίες θα χρειαστούν στην συνέχεια ώστε να γίνουν πιο κατανοητά τα βασικά προβλήματα που η διπλωματική επιχειρεί να επιλύσει. Τέτοιες έννοιες είναι οι βασικοί τύποι κατηγοριοποιητών, να γίνει αντιληπτό το πρόβλημα του σταθερού κ στον αλγόριθμο KNN αλλά και να γίνει κατανοητό το πρόβλημα της απλής παραλλαγής του AdaNN όπως προκύπτει. Επίσης στην εισαγωγή περιλαμβάνεται η παρούσα ενότητα η οποία περιγράφει την βασική διάρθρωση της διπλωματικής. Το κεφάλαιο 2 περιλαμβάνει βασικές γνώσεις για τον κατηγοριοποιητή KNN. Τέτοιες γνώσεις είναι ο τρόπος λειτουργίας του, τα διάφορα μειονεκτήματα και πλεονεκτήματα τα οποία έχει καθώς και την προσπάθεια επίλυσης του βασικού προβλήματος του KNN το οποίο είναι η σταθερή τιμή του κ. Επίσης στο κεφάλαιο 2 γίνονται ανάλυση και κάποιων βασικών παραλλαγών που έχει αλγόριθμος KNN αλλά και κάποιων παραλλαγών των διάφορων μετρικών απόστασης που χρησιμοποιούνται στον αλγόριθμο. Παραδείγματα τέτοιων μετρικών είναι η Manhattan, η Ευκλείδεια και η Minkowski. Στο Κεφάλαιο 3 γίνεται μια ανάλυση των διάφορων παραλλαγών του αλγορίθμου AdaNN που έχουν υλοποιηθεί στην διπλωματική εργασία αλλά και του βασικού αλγορίθμου AdaNN. Επίσης σημαντικό σημείο είναι ότι γίνεται ανάλυση του βασικού προβλήματος του αλγορίθμου AdaNN το οποίο είναι το συγκεκριμένο μέγιστο όριο του βέλτιστου κ που έχει ο αλγόριθμος AdaNN. Στο αμέσως επόμενο κεφάλαιο γίνεται μια ανάλυση στον τρόπο με τον οποίο υλοποιήθηκε ο κώδικας των διάφορων παραλλαγών στην γλώσσα Python, τις όποιες τεχνικές λεπτομέρειες αλλά και ανάλυση των βιβλιοθηκών που χρησιμοποιήθηκαν. Τέλος πριν τα συμπεράσματα είναι η πειραματική μελέτη στην οποία γίνεται ανάλυση των συνόλων δεδομένων, ο τρόπος με τον οποίο έτρεξαν τα πειράματά αλλά και παράθεση των αποτελεσμάτων των πειραματικών μετρήσεων.

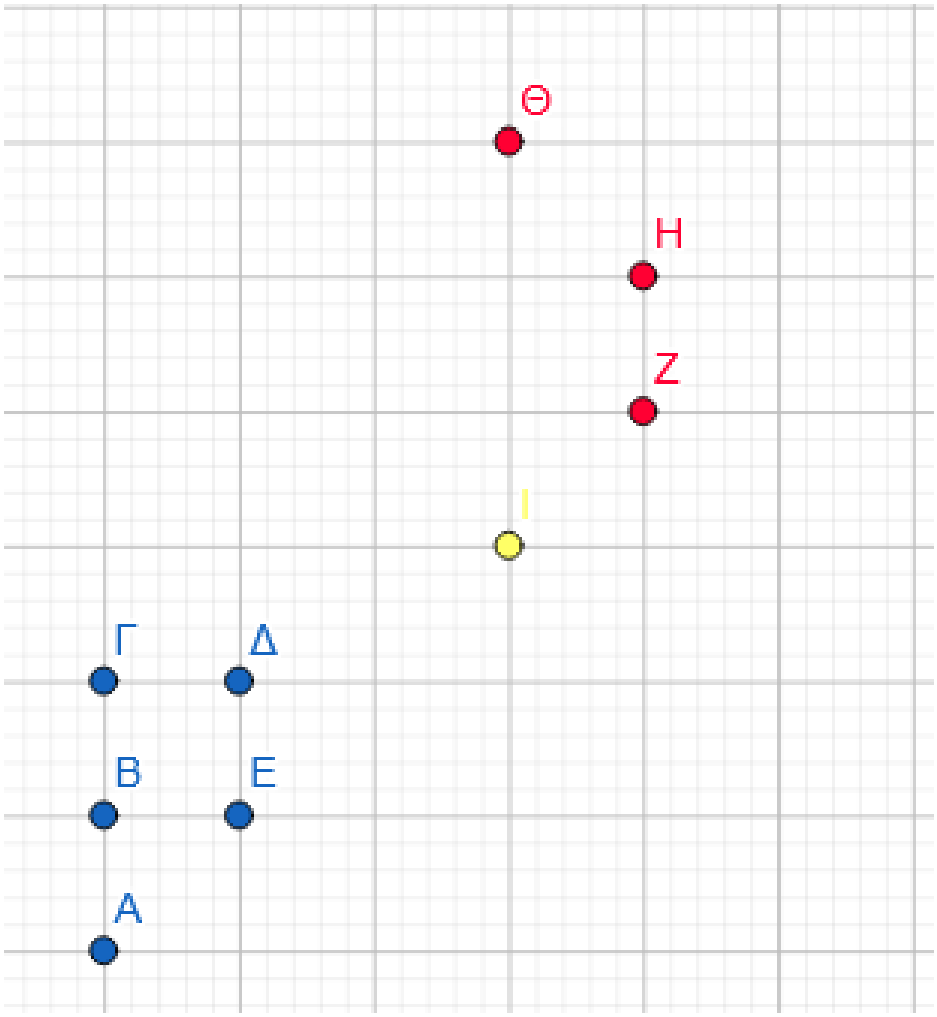
Κεφάλαιο 2

Κατηγοριοποίηση Εγγύτερων Γειτόνων

2.1 Ο Κατηγοριοποιητής Κ Εγγύτερων Γειτόνων

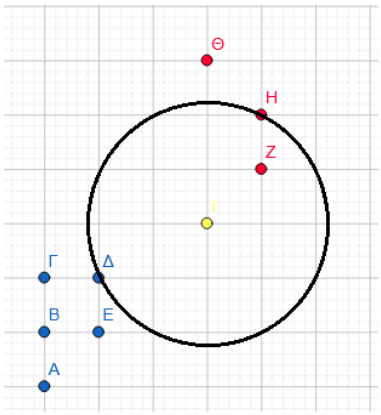
Πριν προχωρήσουμε στην ανάλυση των παραλλαγών του AdaNN που δημιουργήθηκαν στην παρούσα διπλωματική είναι σημαντικό να γίνει μια ανάλυση τόσο στον κατηγοριοποιητή κ εγγύτερων γειτόνων που αποτελεί την βάση του AdaNN αλλά και να αναλυθούν τα επιμέρους ζητήματα που έχουν οι αλγόριθμοι αυτοί.
Ο πρώτος και πιο σημαντικός αλγόριθμος που θα αναλυθεί είναι ο αλγόριθμος κ εγγύτερων γειτόνων. Είναι ιδιαίτερα σημαντικό να αναφέρουμε ότι ο κατηγοριοποιητής Κ εγγύτερων γειτόνων είναι μια βασική παραλλαγή της γενικότερης προσέγγισης εγγύτερων γειτόνων [6]. Η προσέγγιση αυτή θα περιλαμβάνει την δημιουργία ενός πίνακα ο οποίος θα περιλαμβάνει όλα τα στιγμιότυπα των οποίων η κλάση τους θα είναι γνωστή. Η λογική με την οποία θα λειτουργεί η προσέγγιση αυτή για κάθε ένα καινούργιο στιγμιότυπο θα είναι η παρακάτω. Αρχικά για κάθε καινούργιο στιγμιότυπο θα γίνεται υπολογισμός μιας τιμής με βάση την οποία θα υπολογίζουμε την ομοιότητα του υπάρχοντος στοιχείου με το νέο στιγμιότυπο το οποίο έρχεται προς ταξινόμηση. Στην συνέχεια θα πρέπει να υπολογιστεί για το καινούργιο στιγμιότυπο με ποιο στοιχείο από τον υπάρχοντα πίνακα μοιάζει περισσότερο για να του κάνουμε ανάθεση την κατηγορία του υπάρχοντος στιγμιότυπου στο καινούργιο. Στην συνέχεια θα πρέπει εφόσον το καινούργιο στιγμιότυπο ανήκει ήδη σε μια κατηγορία να γίνει πρόσθεσή του στον πίνακα με τα ήδη υπάρχοντα στιγμιότυπα. Αυτή είναι η μέθοδος κατηγοριοποίησης πλησιέστερου γείτονα. Όπως φαίνεται σε αυτή την μέθοδο αντί για να γίνει μια δημιουργία ενός γενικότερου μοντέλου, γίνεται αποθήκευση όλων των στιγμιοτύπων. Για αυτόν τον λόγο και στην προκειμένη περίπτωση του κατηγοριοποιητή K-NN αναφερόμαστε σε έναν οκνηρό αλγόριθμο. Όπως φαίνεται και στο παρακάτω σχήμα 2.1 βλέπουμε ένα καινούργιο στιγμιότυπο προς κατηγοριοποίηση. Η λογική του κατηγοριοποιητή κ εγγύτερων γειτόνων λαμβάνει ως δεδομένο ότι κάθε στιγμιότυπο που έχει ίδια ή παρόμοια δεδομένα εισόδου, θα έχουν και την ίδια έξοδο. Δηλαδή αν δυο σημεία σε έναν πολυδιάστατο χώρο είναι πολύ κοντά τότε θεωρείται ως δεδομένο ότι και οι κατηγορίες τους είναι ίδιες [7]. Ένα χαρακτηριστικό παράδειγμα συνόλου δεδομένων μπορεί να φανεί στο παρακάτω σχήμα σχήμα 2.1 τα σημεία τα οποία αποτελούν το σύνολο δεδομένων είναι τα [Α,Β,Γ,Δ,Ε,Ζ,Η,Θ] και το νέο σημείο ή αλλιώς στιγμιότυπο προς ταξινόμηση είναι το κίτρινο στιγμιότυπο. Τα δεδομένα τα οποία έχουν ίδια ή παρόμοια είσοδο (τετμημένη, τεταγμένη) έχουν και ίδια έξοδο καθώς βλέπουμε ότι τα κοντινά μεταξύ τους σημεία ανήκουν στην μπλε κατηγορία και τα υπόλοιπα κοντινά μεταξύ τους σημεία ανήκουν στην κόκκινη

κατηγορία.



Σχήμα 2.1: Προσέγγιση Ταξινόμησης Εγγύτερων Γειτόνων.

Σε αυτό το σημείο όμως αυτό που πρέπει να απαντηθεί είναι ποια μετρική απόστασης και ομοιότητας θα χρησιμοποιηθεί καθώς και να δούμε ποιοι γείτονες θεωρούνται εγγύτεροι. Η πιο απλή μετρική η οποία χρησιμοποιείται είναι η ευκλείδεια απόσταση, αν και σε πολλά προβλήματα μπορεί να μην είναι η κατάλληλη. Όσον αφορά το ποιοι γείτονες είναι κοντινοί αυτό ορίζεται στον αλγόριθμος K-NN κατά την εφαρμογή του με μια μεταβλητή κ η οποία αποτελεί υπέρ-παράμετρο και θα ορίσει το πόσοι γείτονες γύρω από το εξεταζόμενο στιγμιότυπο θα ερωτηθούν για να βρεθεί η πλειοψηφούσα κλάση σε εκείνη την γειτονία. Οπότε σε μια περίπτωση που αλγόριθμος είναι 3-NN, ο αλγόριθμος θα πάει να βρει τους 3 κοντινότερους γείτονες για να εντοπίσει ποια κατηγορία είναι πιο ισχυρή στην γειτονιά του. Συνεπώς με την αλλαγή της υπερ-παραμέτρου κ ορίζουμε και το εύρος της γειτονιάς του κάθε στιγμιότυπου. Στο παρακάτω παράδειγμα βλέπουμε πως το ίδιο νέο στιγμιότυπο θα εξεταστεί με την εκτέλεσή του αλγορίθμου 3-NN και θα ταξινομηθεί στην κόκκινη κλάση.



Σχήμα 2.2: Προσέγγιση Ταξινόμησης 3 Εγγύτερων Γειτόνων.

2.2 Πλεονεκτήματα και Μειονεκτήματα Κατηγοριοποιητή Κ Εγγυτερων Γειτόνων

Σε αυτό το σημείο είναι ιδιαίτερα σημαντικό να γίνει αναφορά στα διάφορα πλεονεκτήματα αλλά και τα μειονεκτήματα που έχει ο αλγόριθμος KNN [8]. Το πρώτο σημαντικό χαρακτηριστικό το οποίο έχει σαν θετικό ο αλγόριθμος των Κ εγγυτέρων γειτόνων είναι ότι είναι απλός στην υλοποίηση. Αυτός είναι και ο βασικός λόγος που πολλοί νέοι επιστήμονες δεδομένων διδάσκονται αυτόν τον αλγόριθμο. Αυτή του η απλότητα επίσης έχει να κάνει με την έννοια της γειτονιάς και της μεταφοράς του πραγματικού κόσμου. Ο αλγόριθμος αυτός μιμείται τον τρόπο με τον οποίο οι άνθρωποι παίρνουν αποφάσεις με βάση την συμπεριφορά και την επιρροή των ανθρώπων που είναι γύρω τους. Για αυτόν τον λόγο αν όλοι οι άνθρωποι γύρω μας έχουν μια συγκεκριμένη άποψη τείνουμε αυτήν την άποψη να την υιοθετούμε και εμείς. Ένα άλλο σημαντικό χαρακτηριστικό είναι ότι δεν χτίζεται ένα σύνθετο μοντέλο. Η ιδιότητα αυτή βασίζεται στο γεγονός ότι ο αλγόριθμος Κ εγγύτερων γειτόνων είναι ένα οκνηρός κατηγοριοποιητής που δεν θα περιλαμβάνει μια διαδικασία μάθησης. Ένα άλλο χαρακτηριστικό είναι ότι το αποτέλεσμα του αλγόριθμου αυτού ερμηνεύεται εύκολα. Για παράδειγμα έστω ότι ένα στιγμιότυπο έχει κατηγοριοποιηθεί με μια συγκεκριμένη κατηγορία. Για να μπορέσουμε να καταλάβουμε το αποτέλεσμα μπορούμε ακόμα και οπτικά να πάμε και να ελέγξουμε σε τι κλάσεις είναι κατηγοριοποιημένοι οι γείτονες του και να μπορέσουμε να ερμηνεύσουμε το αποτέλεσμα. Αυτό το συγκεκριμένο χαρακτηριστικό μας οδηγεί και στο επόμενο πλεονέκτημα το οποίο έχει να κάνει με την εύκολη αναπαράσταση του αποτελέσματος του αλγορίθμου, δεδομένου ότι ο αλγόριθμος μπορεί να κάνει απεικόνιση και σε δισδιάστατο και τρισδιάστατο χώρο και μπορούν

να ελεγχθούν τα όρια με τα οποία έχει γίνει η λήψη της απόφασης. Το επόμενο χαρακτηριστικό το οποίο θα αναφερθεί έχει να κάνει με τη ευελιξία του αλγορίθμου KNN. Η ευελιξία αυτή έχει να κάνει με το γεγονός ότι μπορεί να χρησιμοποιηθεί και για προβλήματα κατηγοριοποίησης και για προβλήματα παλινδρόμησης. Ένα επίσης σημαντικό χαρακτηριστικό είναι ότι μπορεί να προσαρμόζεται εύκολα σε νέα δεδομένα. Όπως και έχει αναφερθεί και πιο πριν στην ενότητα των ειδών κατηγοριοποιητών ο αλγόριθμος KNN είναι οκνηρός κατηγοριοποιητής πράγμα το οποίο σημαίνει ότι δεν χτίζει μοντέλο αλλά προσαρμόζεται στα καινούργια δεδομένα τα οποία έρχονται. Όσο έρχονται δεδομένα ο KNN θα κάνει προσαρμογή στα δεδομένα που θα αποθηκευτούν στην μνήμη. Ένα επίσης σημαντικό πλεονέκτημα είναι ότι παίρνει πολύ λίγες υπέρ-παραμέτρους. Για να εκτελεστεί ο αλγόριθμος K εγγύτερων γειτόνων δεν θα χρειαστούν πολλές παράμετροι, αλλά μόνο η τιμή του K καθώς και η μετρική απόστασης με την οποία θα υπολογιστεί το μέτρο της ομοιότητας των στιγμιοτύπων.

- **Απλότητα**
- **Μεταφορά Πραγματικού Κόσμου**
- **Απουσία Σύνθετου Μοντέλου**
- **Εύκολη Ερμηνεία και Οπτική Αναπαράσταση**
- **Ευελιξία**
- **Λίγες Υπέρ-παράμετροι**

Σε αυτό το σημείο είναι πολύ σημαντικό να αναφερθούμε και στα μειονεκτήματα τα οποία έχει ο αλγόριθμος KNN τα οποία θα ήταν χρήσιμο να μπορέσουν να βελτιωθούν. Το ένα πρόβλημα έχει να κάνει με την κλιμάκωση του KNN. Όπως έχει αναφερθεί ο αλγόριθμος K εγγύτερων γειτόνων δεν έχει μοντέλο. Αυτό σημαίνει ότι όλα τα στιγμιότυπα τα οποία θα χρησιμοποιήσει τα έχει αποθηκευμένα στην μνήμη. Άρα έχει το μειονέκτημα της χρήσης περισσότερης μνήμης και αποθηκευτικού χώρου το οποίο θα έχει άμεσες επιπτώσεις και σε χρόνο και σε χρήμα στην εταιρία ή τον οργανισμό που θα χρησιμοποιήσει τον αλγόριθμο. Αυτό συμβαίνει γιατί όσο θα αυξάνονται τα δεδομένα, τόσο θα αυξάνεται και ο χρόνος των υπολογισμών. Για αυτόν τον λόγο και έχουν υιοθετηθεί διαφορετικές δομές δεδομένων όπως ο Ball Tree και το Kd Tree τα οποία θα συζητηθούν παρακάτω. Αυτό το πρόβλημα της κλιμάκωσης έχει να κάνει και με την κλιμάκωση του αριθμού των στιγμιοτύπων, αλλά και την κλιμάκωση του αριθμού ενός χαρακτηριστικού. Το πρώτο έχει να κάνει με την αύξηση των στιγμιοτύπων που μπορεί να προκύψουν σε περίπτωση μεγάλων συνόλων δεδομένων και το δεύτερο έχει να κάνει με την μη αποκλιμάκωση ενός χαρακτηριστικού που έχει ένα στιγμιότυπο. Ένα χαρακτηριστικό παράδειγμα είναι αν ένα στιγμιότυπο έχει δύο χαρακτηριστικά και το ένα είναι το ύψος ενός ανθρώπου και το δεύτερο είναι τα κιλά του ο αλγόριθμος μπορεί να νομίζει ότι ο αριθμός των κιλών είναι πιο σημαντικό χαρακτηριστικό διότι έχει πολύ μεγαλύτερες τιμές. Αυτή η μη αποκλιμάκωση του συγκεκριμένου χαρακτηριστικού μπορεί να οδηγήσει σε πόλωση (bias) στον αλγόριθμό μας και να έχει αρνητικό αντίκτυπο στην απόδοση του. Το πρόβλημα της αποκλιμάκωσης μπορεί να λυθεί με την κανονικοποίησης της τιμής αυτής. Ένας τρόπος είναι να γίνει μια μετατροπή σε αφαιρώντας από την μεγαλύτερη τιμή που έχει σε αυτό το χαρακτηριστικό όλο το σύνολο δεδομένων την αντίστοιχη μικρότερη τιμή. Ένας άλλος τρόπος αντιμετώπισης της κλιμάκωσης των δεδομένων είναι η κατάλληλη επιλογή της απόστασης μεταξύ

των στιγμιοτύπων, εφόσον η κλασσική ευκλείδεια απόσταση μπορεί να μην είναι κατάλληλη για το συγκεκριμένο σύνολο δεδομένων. Ένά άλλο σημείο το οποίο πρέπει να αναφερθεί έχει να κάνει και με την κλιμάκωση του αριθμού των χαρακτηριστικών. Το πρόβλημα αυτής της ύπαρξης των πολλών διαφορετικών διαστάσεων θα έχει άμεσο αντίκτυπο στον χρόνο εκτέλεσης αλλά και στον αριθμό των δεδομένων τα οποία υπάρχουν αποθηκευμένα στην μνήμη. Για αυτό τον λόγο είναι πιθανό να χρειαστεί μια τεχνική μείωσης των διαστάσεων, όπως η PCA (Principal Componenent Analysis). Τέλος όσον αφορά την κλιμάκωση των δεδομένων υπάρχει και το πρόβλημα του κ. Αυτό έχει να κάνει με την κατάλληλη επιλογή της τιμής του ώστε και να μην υπάρξει υπό-εκπαίδευση αλλά ούτε και υπέρ-εκπαίδευση. Ένα από τα προβλήματα της κλιμάκωσης το οποίο έχει αναφερθεί έχει να κάνει και με την διαχείριση των πολλών διαστάσεων. Σε αυτό το σημείο είναι άξιο αναφοράς ότι και ο αλγόριθμος κ εγγύτερων γειτόνων έχει ως μειονέκτημα την κατάρα των πολλών διαστάσεων όπως και πολλοί άλλοι αλγόριθμοι. Αυτό στην πορεία θα αποτελέσει και θέμα στην συζήτηση όσον αφορά τον τρόπο με τον οποίο θα βρεθεί το K με το οποίο θα χρησιμοποιήσουμε στον αλγόριθμο. Επίσης σημαντικό πρόβλημα σε σχέση με τον αλγόριθμο KNN είναι η διαχείριση των ισοτήτων των κλάσεων σε μια γειτονιά. Για παράδειγμα έστω ότι ένα στιγμιότυπο πάει προς εξέταση και προσπαθεί να βρει την πλειοψηφούσα κλάση στην γειτονιά του και βρίσκει παραπάνω από μια κλάση που να έχουν τον μέγιστο αριθμό στιγμιοτύπων στην γειτονιά. Μια αρχική λύση που θα μπορούσε κάποιος να εντοπίσει είναι να δοθεί περιττός αριθμός κ για να υπάρχει μονός αριθμός στιγμιοτύπων στην γειτονιά. Αυτή η λύση όμως δεν μπορεί να γενικεύει για προβλήματα με παραπάνω από 2 κατηγορίες.

- **Κλιμάκωση Αριθμού Στιγμιοτύπων**
- **Κλιμάκωση Αριθμού Διαστάσεων**
- **Κλιμάκωση Τιμής Χαρακτηριστικού**
- **Υψηλή Χρησιμοποίηση Μνήμης**
- **Υψηλή Χρησιμοποίηση Αποθηκευτικού Χώρου**
- **Κατάρα των πολλών διαστάσεων**
- **Κατάλληλη Επιλογή του κ**
- **Διαχείριση Ισοτήτων**

2.3 Προσδιορισμός της παραμέτρου κ

Σε αυτό το σημείο είναι πολύ σημαντικό να αναφερθούμε στον τρόπο επιλογής της παραμέτρου κ η οποία θα χρειαστεί για την εκτέλεση του αλγορίθμου K-NN. Το να βρεθεί η κατάλληλη τιμή για το κ για τον αλγόριθμο κ εγγύτερων γειτόνων είναι μια σύνθετη διαδικασία και περιλαμβάνει μια διαδικασία που λέγεται tuning της υπέρ-παραμέτρου κ. Ο στόχος ο οποίος υπάρχει είναι να βρεθεί μια τιμή κ ώστε ο αλγόριθμος K-NN να έχει την μεγαλύτερη απόδοση σε νέα καινούργια δεδομένα. Η τιμή του κ η οποία θα επιλεχθεί είναι ο αριθμός των γειτόνων που θα

ελέγξει ένα στιγμιότυπο για να βρει την πλειοψηφούσα κλάση. Συνεπώς ο επιλεγμένος αριθμός κ είναι ένας θετικός αριθμός. Συγκεκριμένα υπάρχουν διάφοροι τρόποι και παράγοντες οι οποίοι θα μας βοηθήσουν να επιλέξουμε το κατάλληλο κ παρουσιάζονται παρακάτω:

- **Τιμές κ**
- **Γνώση Domain και Πειραματική Αξιολόγηση**
- **Cross-Validation**

2.3.1 Τιμές κ

Το πρώτο σημείο στο οποίο πρέπει να σταθούμε είναι οι πιθανές τιμές οι οποίες μπορούν να έρθουν σαν υπέρ-παράμετροι κ στον αλγόριθμο κ εγγύτερων γειτόνων καθώς και το αντίκτυπό τους. Οι τιμές του κ μπορούν να είναι μικρές, μεσαίες και μεγάλες ανάλογα βέβαια και με το σύνολο των δεδομένων. Θα μπορούσαμε να πούμε μικρές τιμές του κ αυτές που έχουν τιμές από 1 έως 3 οι οποίες θα έχουν ως αντίκτυπο την δημιουργία μικρότερων γειτονιών. Αυτό μπορεί να έχει ως αποτέλεσμα την δημιουργία πιο γρήγορης και ευέλικτης γενίκευσης η οποία είναι πιο ευαίσθητη στον θόρυβο. Έστω για παράδειγμα ότι πάει να κατηγοριοποιηθεί ένα στιγμιότυπο μιας κλάσης που στην ευρεία του περιοχή είναι πιο πολλά τα στιγμιότυπα της κλάσης της οποίας πραγματικά ανήκει αλλά να κατηγοριοποιηθεί λάθος αν υπάρχουν κοντά του στιγμιότυπα που είναι θόρυβος. Από την άλλη στις πολύ μεγαλύτερες τιμές του κ μπορεί να υπάρχει ανοχή στον θόρυβο και στα μακρινά στιγμιότυπα αλλά να μην είναι τόσο αποδοτικές στο να ανιχνεύσουν ιδιαιτερότητες της γειτονιάς που βρίσκεται ένα στιγμιότυπο. Για αυτόν τον λόγο μια μεσαία τιμή του κ μπορεί να αποτελέσει μια πολύ καλή αρχή για την εξερεύνηση της καλύτερης τιμής του κ για το σύνολο δεδομένων στο οποίο εφαρμόζεται ο αλγόριθμος. Σημαντικό να αναφερθεί είναι πως οι έννοιες του μικρού, μεγάλου και μεσαίου κ εξαρτώνται πάρα πολύ από το είδος των δεδομένων που υπάρχουν αλλά και από το μέγεθος του συνόλου δεδομένων.

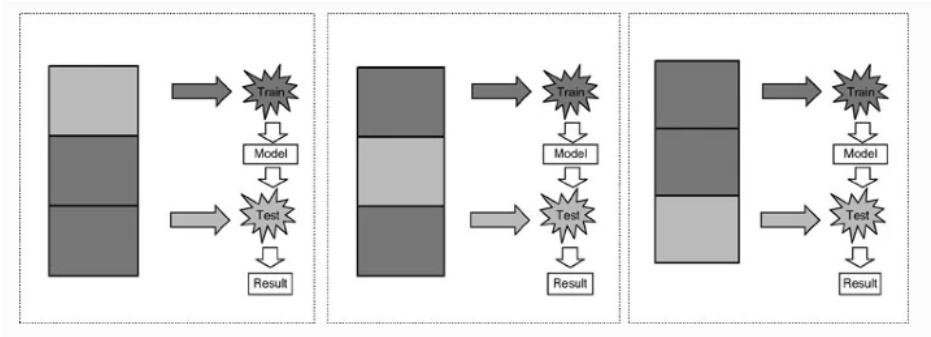
2.3.2 Γνώση Domain και Πειραματική Αξιολόγηση

Σε αυτό το σημείο θεωρείται κατάλληλο να γίνει μια ανάλυση στην σημαντικότητα της γνώσης που πρέπει να υπάρχει πάνω στο σύνολο των δεδομένων. Το πρώτο και πιο σημαντικό σημείο είναι το μέγεθος του συνόλου δεδομένων. Για παράδειγμα μπορεί σε πιο μικρά σύνολα δεδομένων να χρειάζεται μικρότερη τιμή του κ λόγω της ανάγκης ανίχνευσης διαφοροποίησης διάφορων προτύπων σε μικρότερες γειτονιές. Αυτό δεν συμβαίνει σε μεγαλύτερα σύνολα δεδομένων διότι θεωρούνται θόρυβος και καλύπτονται από την ισχύ της επικρατούσας κλάσης σε μια μεγαλύτερη περιοχή. Επίσης σημαντική είναι η γνώση γενικά του συνόλου δεδομένων καθώς μπορεί να υπάρχει μια εικόνα της κλίμακας των μεγεθών των πιθανών γειτονιών που θα χρειαστούν κατά την εφαρμογή του αλγορίθμου κ εγγύτερων γειτόνων. Για αυτόν τον λόγο η πρώτη μέθοδος αναζήτησης του καλύτερου κ είναι με βάση τα πειράματα τα οποία κάνουμε πάνω στο σύνολο δεδομένων με την λογική της δοκιμής ενός πιθανού κ και της αξιολόγησης των αποτελεσμάτων αυτού του κ από τις μετρικές αξιολόγησης των αλγορίθμων.

2.3.3 Εύρεση Βέλτιστης Παραμέτρου k

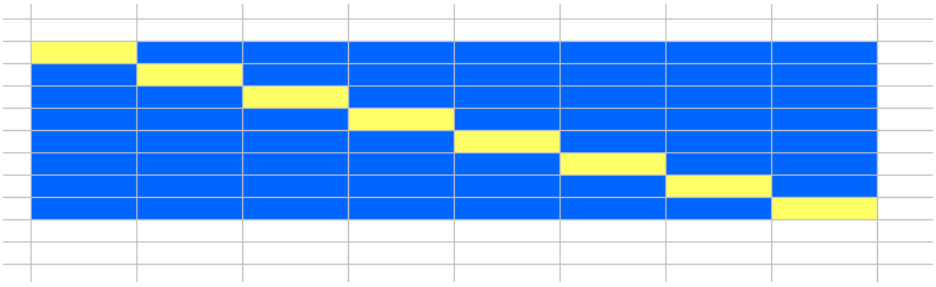
Σε αυτό το σημείο θα πρέπει να αναλυθεί πως για το tuning της παραμέτρου κ χρησιμοποιείται η μέθοδος Cross Validation η οποία είναι μια από τις πιο σημαντικές αυτόματες τεχνικές αναζήτησης του βέλτιστου κ για το τρέχον σύνολο δεδομένων [9]. Το Cross Validation είναι μια μέθοδος η οποία χρησιμοποιείται στην στατιστική για να μπορούμε να συγκρίνουμε αλγορίθμους και να τους αξιολογήσουμε. Η μέθοδος αυτή χρειάστηκε επειδή είναι πολύ πιθανό μετά από μια εκπαίδευση ενός αλγορίθμου σε κάποια συγκεκριμένα δεδομένα και επιτυχημένης επαλήθευσης του αλγορίθμου, ο αλγόριθμος να αποτυγχάνει όταν θα έρθουν καινούρια δεδομένα. Στην τακτική αυτή υπάρχει διαχωρισμός των δεδομένων σε δύο μέρη. Το πρώτο μέρος λέγεται σύνολο εκπαίδευσης του αλγορίθμου και το άλλο λέγεται σύνολο επαλήθευσης του αλγορίθμου. Η κλασσική παραλλαγή της μεθόδου αυτής είναι το k fold cross-validation αλλά υπάρχουν και άλλες παραλλαγές όπως το Leave-One-Out Cross Validation. Ο πρώτος και πιο σημαντικός αλγόριθμος που θα αναλυθεί είναι ο k fold cross-validation. Η λογική της λειτουργίας της τεχνικής περιλαμβάνει την κατάτμηση του συνόλου δεδομένων σε k αριθμο folds τα οποία είναι ίσα σε μέρη. Στην συνέχεια γίνονται κ επαναλήψεις όπου σε κάθε μία γίνεται εκπαίδευση και επαλήθευση του μοντέλου. Η λογική είναι ότι ένα κομμάτι θα χρησιμοποιηθεί για την εκπαίδευση του μοντέλου π.χ. στην εκπαίδευση θα χρησιμοποιηθούν τα 8 από τα 10 κομμάτια και το κομμάτι το οποίο απομένει θα χρησιμοποιηθεί στην επαλήθευση του μοντέλου. Στο παρακάτω σχήμα 2.3 βλέπουμε ένα παράδειγμα από Cross Validation το οποίο εκτελείται με 3 διαφορετικά folds.

Στο παρακάτω σχήμα βλέπουμε πως στην πρώτη επανάληψη το δεύτερο και το τρίτο μέρος θα αποτελέσουν το σύνολο της εκπαίδευσης και το πρώτο θα χρησιμοποιηθεί για την επαλήθευση του μοντέλου. Στην δεύτερη επανάληψη βλέπουμε πως το πρώτο και το τρίτο μέρος χρησιμοποιούνται για την εκπαίδευση του αλγορίθμου και το δεύτερο θα χρησιμοποιηθεί τον έλεγχο της επίδοσης του αλγορίθμου. Στην τρίτη και τελευταία φάση του cross validation γίνεται εκπαίδευση με τα πρώτα δυο μέρη του συνόλου δεδομένων και το τρίτο χρησιμοποιείται για την επαλήθευση της ικανότητας του μοντέλου να γενικεύσει. Άρα μετά από αυτό το παράδειγμα μπορούμε να περιγράψουμε την γενική μορφή του κανόνα του k fold cross-validation όπου χωρίζουμε τα δεδομένα μας σε k - 1 folds για την δημιουργία διαφορετικών μοντέλων και δημιουργία εκτιμήσεων με βάση κάποια μετρική απόδοσης. Μετά από την εκτέλεση όλων των k folds θα γίνει μια σύνοψη όλων των εκτιμήσεων όλων των folds έτσι ώστε να γίνει εκτίμηση της ικανότητας γενίκευσης του αλγορίθμου με διαφορετικά δεδομένα κάθε φορά. Στην περίπτωση της επιλογής του καλύτερου κ θα πρέπει για αρχή να γίνει ένας καθορισμός των τιμών του κ όπου θα επιχειρηθεί η αναζήτηση. Ένα παράδειγμα θα μπορούσε να είναι η εκτέλεση με ελάχιστο αριθμό κ το 1 και ως μέγιστο το 30. Στην συνέχεια θα πρέπει να γίνει χωρισμός σε k folds και θα πρέπει να γίνει εκτίμηση όλων των μοντέλων για κάθε τιμή κ. Μετά την εκτέλεση του cross validation θα πρέπει να γίνει επιλογή αυτού του κ που είχε ως αποτέλεσμα την καλύτερη απόδοση στην επιλεχθείσα μετρική. Γενικά οι στόχοι του cross validation είναι να γίνει μια εκτίμηση της ικανότητας του αλγορίθμου να γενικεύει σε διαφορετικά είδη δεδομένων. Ένα άλλος στόχος που μπορεί να έχει το cross validation είναι η σύγκριση της απόδοσης δύο διαφορετικών αλγορίθμων ή σε κάποια άλλη περίπτωση να γίνει σύγκριση 2 παραμέτρων του ίδιου αλγορίθμου.
Η αμέσως επόμενη παραλλαγή του cross validation είναι το Leave-One-Out 2.4. Στην συγκεκριμένη παραλλαγή βλέπουμε μια μέθοδο η οποία είναι μια ειδική περίπτωση του k fold cross



Σχήμα 2.3: Μια απλή εφαρμογή Cross Validation [9].

validation. Στην συγκεκριμένη παραλλαγή ο αριθμός των folds ισούται με τον αριθμό των στιγμιοτύπων που υπάρχουν στο σύνολο δεδομένων. Από όλα αυτά τα στιγμιότυπα όλα εκτός από ένα θα χρησιμοποιηθούν για την εκπαίδευση του αλγορίθμου και η τελευταία παρατήρηση θα χρησιμοποιηθεί για τον έλεγχο της επίδοσης του αλγορίθμου.



Σχήμα 2.4: Εφαρμογή Leave One Out Cross Validation.

2.4 Μετρικές απόστασης

Σε αυτό το σημείο εφόσον έχουμε μιλήσει και για τον κατηγοριοποιητή K εγγύτερων γειτόνων είναι ανάγκη να γίνει αναφορά και στις μετρικές αποστάσεις οι οποίες χρησιμοποιούνται στους αλγόριθμους εξόρυξης δεδομένων και στον K-NN [10]. Πολλές φορές κατά την διάρκεια της παρούσας διπλωματικής γίνεται αναφορά στην εγγύτητά δύο στιγμιοτύπων ή ότι ένα στιγμιότυπο είναι ”κοντά” σε ένα άλλο ή ”κοντά” σε κάποια κλάση. Σε αυτό το σημείο θα γίνει μια προσπάθεια για να παρουσιαστεί ο τρόπος με τον οποίο υπολογίζεται η εγγύτητά. Αυτό το οποίο είναι σημαντικό να γίνει αντιληπτό είναι πως λαμβάνουμε ως δεδομένο ότι το κριτήριο της απόστασης λογίζεται σαν μετρική ομοιότητας μεταξύ δύο ή παραπάνω στιγμιοτύπων. Αυτό συμβαίνει με την λογική ότι κάθε στιγμιότυπο μπορεί να απεικονιστεί στον χώρο σαν ένα σημείο με τόσες συντεταγμένες όσες και τα χαρακτηριστικά που έχει ένα στιγμιότυπο. Συνεπώς και όσο πιο πολλά χαρακτηριστικά έχει ένα στιγμιότυπο τόσες διαστάσεις θα χρειαστεί αρχικά για να απεικονιστεί και στην συνέχεια να υπολογιστεί η εγγύτητα με άλλα στιγμιότυπα του χώρου. Δεδομένου ότι κάθε ένα στιγμιότυπο αποτελείται από έναν αριθμό χαρακτηριστικών, άμα ένα στιγμιότυπο έχει παραπλήσιες τιμές χαρακτηριστικών με ένα άλλο στιγμιότυπο

τότε θεωρούμε ότι έχουμε πολύ μικρή απόσταση μεταξύ τους και συνεπώς πολύ μεγαλύτερο αριθμό ομοιότητας μεταξύ των στιγμιοτύπων. Σε αυτό το σημείο είναι αναγκαίο να γίνει αντιληπτό πως η ομοιότητα είναι υποκειμενική και πως θα πρέπει να λαμβάνουμε υπόψη τις άλλες παραμέτρους του προβλήματος. Ένα τέτοιο παράδειγμα είναι η ομοιότητα των χρωμάτων των μαλλιών δύο ανθρώπων. Άρα σε αυτό το σημείο κατά την φάση της προ-επεξεργασίας θα πρέπει να γίνει κάποια μετατροπή των χαρακτηριστικών του στιγμιότυπου ώστε να είναι συγκρίσιμα. Σε αυτό το σημείο θα πρέπει να αναφερθεί πως και ο K-NN χρειάζεται τιμές σαν χαρακτηριστικά και όχι συμβολοσειρές. Χαρακτηριστικά παραδείγματα μετρικών τα οποία πρέπει να αναφερθούν είναι η ευκλείδια απόσταση, η απόσταση Manhattan, η απόσταση Jaccard, η απόσταση Minkowski και η απόσταση Cosine

- **Ευκλείδεια απόσταση**
- **Απόσταση Manhattan**
- **Απόσταση Jaccard**
- **Απόσταση Minkowski**
- **Απόσταση Cosine**
- **Απόσταση Hamming**

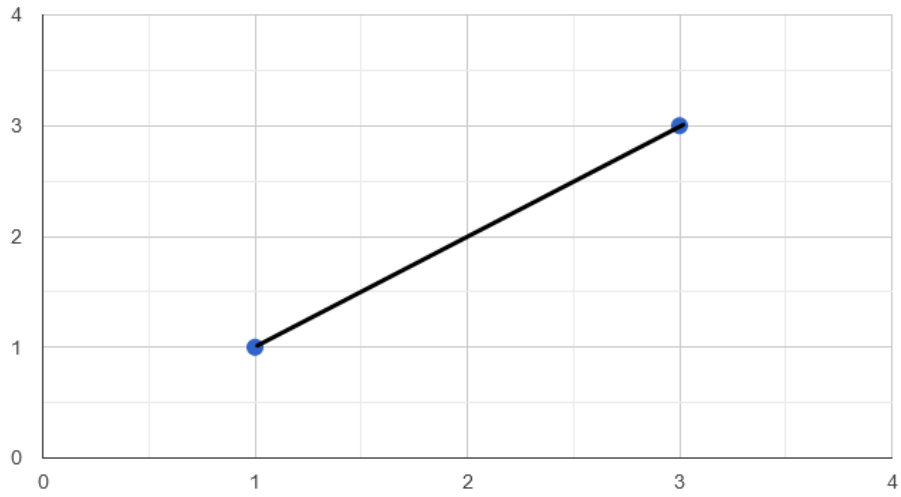
Το πρώτο είδος μετρικής απόστασης που θα αναλυθεί είναι η Ευκλείδεια απόσταση. Η Ευκλείδεια απόσταση είναι μια μετρική η οποία παραδοσιακά χρησιμοποιείται στα προβλήματα γεωμετρίας και είναι η πιο συνηθισμένη απόσταση η η οποία χρησιμοποιείται και στον πραγματικό κόσμο. Η μετρική αυτή χρησιμοποιείται από μια πληθώρα αλγορίθμων με γνωστά παραδείγμα αυτών να είναι ο K-NN και ο K-means. Αυτό το οποίο υπολογίζει είναι ρίζα της διαφοράς τετραγώνων των συντεταγμένων οι οποίες ισχύουν για δυο στιγμιότυπα για να υπολογιστεί η απόσταση η οποία αλλιώς ονομάζεται νόρμα. Η Ευκλείδεια απόσταση μπορεί να υπολογιστεί από τον παρακάτω τύπο.

```
||x, y| = √((x1 − y1)^2 + (x2 − y2)^2)
```

Το δεύτερο είδος μετρικής απόστασης που θα αναλυθεί είναι η Manhattan η οποία υπολογίζεται από τις διαφορές συντεταγμένων των στιγμιοτύπων σε απόλυτη τιμή. Η μετρική αυτή λέγεται αλλιώς και μετρική του ταξί. Αυτό συμβαίνει καθώς η μετρική αυτή παρομοιάζεται με την διαδρομή που κάνουν τα ταξί σε μια πόλη. Με την ίδια λογική που ένα ταξί δεν μπορεί να ακολουθήσει την συντομότερη δυνατή διαδρομή η οποία είναι η Ευκλείδεια, το ταξί ακολουθεί κινήσεις οριζόντιες και κάθετες [11].

```
||x − y|| = |x1 − y1| + |x2 − y2|
```

Το τρίτο είδος μετρικής απόστασης που θα αναφερθεί είναι η Jaccard, όπου μιλάμε για ομοιότητα συνόλων δεδομένων. Η απόσταση Jaccard μετράει την ομοιότητα μεταξύ δύο συνόλων με την εξής λογική. Αρχικά βρίσκει την τομή των δύο στοιχείων και μετά την διαιρεί με το πλήθος των στιγμιοτύπων που βρίσκονται στην ένωση των δύο συνόλων. Συνεπώς θα πρέπει πρώτα από όλα να βρει τα στιγμιότυπα που ανήκουν στο σύνολο A, στην συνέχεια τα στιγμιότυπα που



Σχήμα 2.5: Η Ευκλείδεια απόσταση γραφικά.

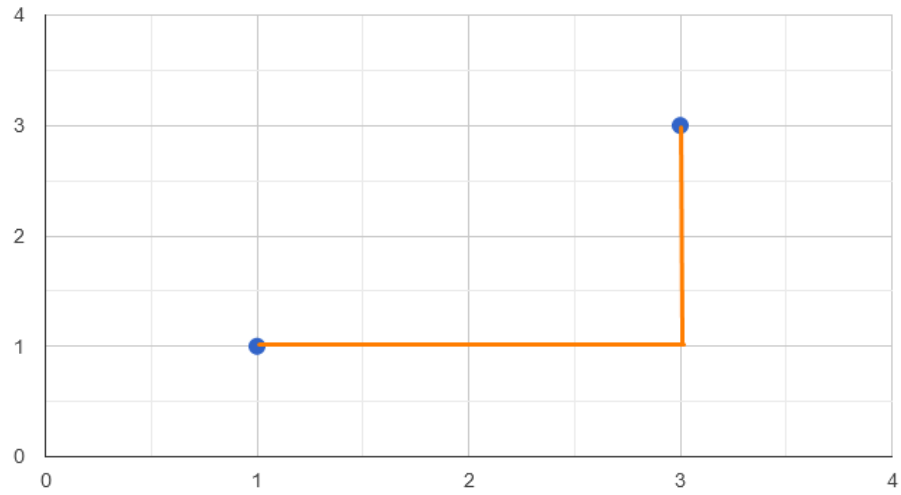
ανήκουν στο σύνολο B και τέλος να υπολογιστεί η τομή των 2 συνόλων. Είναι σημαντικό να γίνει κατανοητό ότι στις πράξεις για τον υπολογισμό αυτό χρησιμοποιούμε τις πληθικότητες των συνόλων.

```
Jaccard_similarity(X, Y) = |X ∩ Y| / |X ∪ Y|
```

Η τέταρτη μετρική που θα αναλυθεί είναι η Minkowski. Αυτή η μετρική αποτελεί μια γενίκευση των 2 περιπτώσεων Ευκλείδειας απόστασης και Manhattan και επιτρέπει τον υπολογισμό μεταξύ δυο σημείων σε ένα πολυδιάστατο χώρο. Ο αλγόριθμος της μετρικής Minkowski ορίζει μια παράμετρο r με βάση την οποία θα γίνει ο υπολογισμός. Όταν η τιμή της μεταβλητής r ισούται με 1 είναι ισοδύναμη με την απόσταση Manhattan και όταν η τιμή r ισούται με 2 είναι ισοδύναμη με την Ευκλείδεια απόσταση. Σε περίπτωση που το r τείνει στο άπειρο τείνει στην απόσταση Chebyshev.

```
||x - y|| = (|x1 - y1|^r + |x2 - y2|^r)^(1/r)
```

Η πέμπτη μετρική απόστασης που θα αναλυθεί είναι η Hamming η οποία μετράει την ομοιότητα μεταξύ 2 συμβολοσειρών οι οποίες και οι 2 έχουν το ίδιο μήκος. Η απόσταση αυτή υπολογίζεται με βάση τους διαφορετικούς τους χαρακτήρες. Το πρώτο πράγμα το οποίο εξετάζεται στην απόσταση Hamming είναι αν οι συμβολοσειρές έχουν το ίδιο μέγεθος. Στην συνέχεια θα γίνει εξέταση ένα προς ένα των γραμμάτων για να δουν ανά θέση αν είναι ίδια. Το τελικό αποτέλεσμα είναι ένα αριθμός ο οποίος δηλώνει πόσα ίδια γράμματα έχουν. Άρα λόγω αυτής της συνθήκης το τελικό αποτέλεσμα είναι ένας αριθμός ο οποίος δεν είναι αρνητικός. Είναι χρήσιμο να αναφερθεί ότι η απόσταση Hamming μπορεί να χρησιμοποιηθεί και για την σύγκρισή συμβολοσειρών με δυαδικά ψηφία ή κοινούς χαρακτήρες που εμπεριέχονται σε ένα αλφάβητο. Γενικότερα η απόσταση αυτή χρησιμοποιείται σε διάφορους τομείς όπως η βιοπληροφορική και η θεωρία της πληροφορίας. Η εφαρμογή της απόστασης αυτή συνήθως χρειάζεται σε διάφορους αλγορίθμους για να γίνει ανίχνευση σφαλμάτων.



Σχήμα 2.6: Η απόσταση Manhattan γραφικά.

```
Hamming_distance(p, q) = number of positions where p_i ≠ q_i
```

Η έκτη μετρική απόστασης που θα αναλυθεί είναι η Cosine. Η μετρική αυτή χρησιμοποιείται σαν κριτήριο ομοιότητας δυο διανυσμάτων. Η απεικόνισή πολλές φορές δύο στιγμιοτύπων μπορεί να γίνει με την μορφή διανυσμάτων. Τέτοια προβλήματα που υπάρχουν στην εξόρυξη δεδομένων έχουν να κάνουν κυρίως με ανάλυση κειμένου αλλά και προβλήματα που έχουν να κάνουν με την επεξεργασία φυσικής γλώσσας. Ο υπολογισμός της ομοιότητας έχει να κάνει με τον υπολογισμό της γωνίας που δημιουργείται από αυτά τα δύο διανύσματα σε έναν διανυσματικό χώρο. Παρακάτω βλέπουμε τον τύπο υπολογισμού της ομοιότητας δύο αντικειμένων.

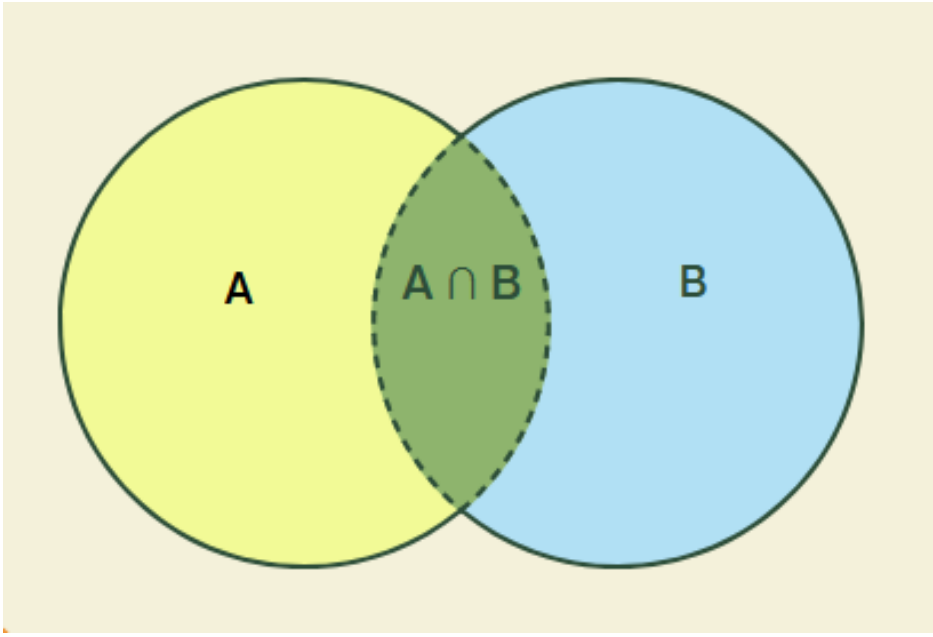
```
cosine_similarity(p, q) = (p · q) / (‖p‖ * ‖q‖)
```

2.5 Παραλλαγές του κατηγοριοποιητή Κ Εγγυτερων Γειτόνων

Σε αυτό το σημείο είναι ιδιαίτερα σημαντικό να αναλυθούν και οι ήδη υπάρχοντες αλγόριθμοι και οι παραλλαγές της μεθόδου κ εγγύτερων γειτόνων [12]. Ο αλγόριθμος K-NN χρησιμοποιείται ευρέως σε προβλήματα της εξόρυξης πληροφορίας, ωστόσο ανάλογα με το πεδίο του προβλήματος μπορεί να χρειάζεται κάποιες διαφοροποιήσεις. Διάφορες παραλλαγές του αλγορίθμου κ εγγύτερων γειτόνων είναι οι εξής: Ο Weighted K-NN, ο KD Tree KNN, ο Ball Tree KNN, ο Bayesian KNN και DB KNN. Όπως έχει αναφερθεί και πιο πάνω ο αλγόριθμος KNN έχει διάφορα μειονεκτήματα τα οποία έχουν γίνει προσπάθειες να προσπεραστούν οι οποίες επικεντρώνονται στις παραλλαγές που θα αναλυθούν παρακάτω.

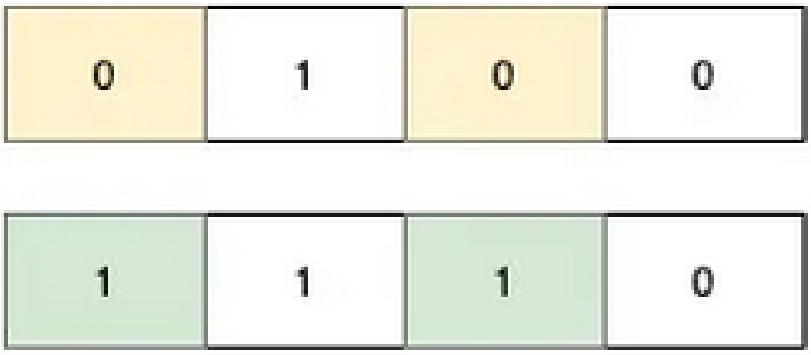
2.5.1 Weighted KNN

Ο πρώτος αλγόριθμος ο οποίος θα αναφερθεί είναι ο Weighted KNN [12][13]. Στον κλασσικό αλγόριθμο KNN όλα τα χαρακτηριστικά έχουν την ίδια σημαντικότητα. Σε κάθε στιγμιότυπο

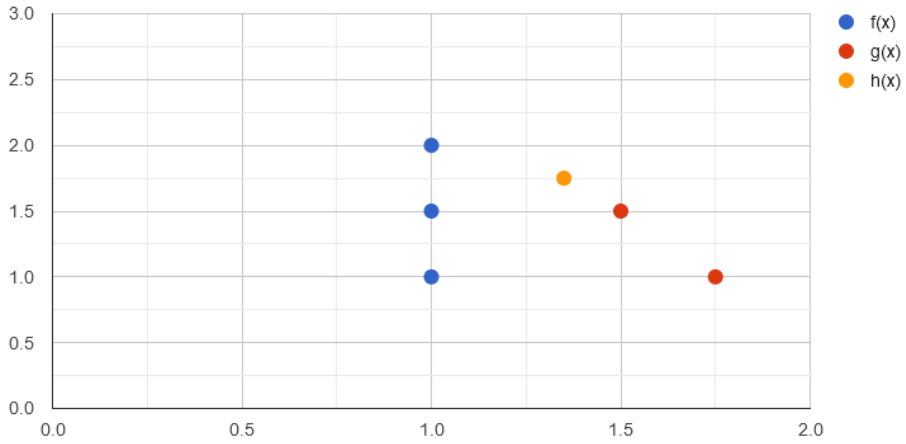


Σχήμα 2.7: Η απόσταση Jaccard γραφικά.

υπάρχει μια σειρά χαρακτηριστικών με βάση τα οποία θα υπολογιστεί η ομοιότητα των στιγμιοτύπων. Στον υπολογισμό της απόστασης μεταξύ των στιγμιοτύπων, κάθε χαρακτηριστικό θα συμβαίνει με την ίδια βαρύτητα. Σε ένα σύνολο δεδομένων όμως τα χαρακτηριστικά τα οποία έχουν τα στιγμιότυπα δεν είναι απαραίτητο να έχουν την ίδια βαρύτητα. Ο αλγόριθμος Weighted KNN δεν θα δίνει ίση βαρύτητα σε κάθε χαρακτηριστικό των στιγμιοτύπων, αλλά θα ακολουθεί μια διαδικασία μάθησης των βαρών για το κάθε ένα χαρακτηριστικό. Στην συνέχεια κατά την διαδικασία της γενίκευσης και εξαγωγής αποτελέσματος, κάθε ιδιότητά του θα επηρεάσει το αποτέλεσμά ανάλογα με την βαρύτητα που έχει πραγματικά το χαρακτηριστικό. Σε πολλές περιπτώσεις μπορεί να δούμε κάποιο χαρακτηριστικό προς εξέταση το οποίο να ταξινομηθεί λάθος λόγω του γεγονότος ότι όλα τα χαρακτηριστικά έχουν την ίδια βαρύτητα. Συγκεκριμένα υπάρχουν πολλές περιπτώσεις όπου ένα στιγμιότυπο μπορεί να είναι πιο κοντά σε ένα άλλο στιγμιότυπο από την κλάση που πραγματικά ανήκει όμως λόγω της πλειοψηφίας των γειτόνων του να ταξινομηθεί σε λάθος κατηγορία. Όπως φαίνεται στο σχήμα 2.9 βλέπουμε ένα πορτοκαλί χαρακτηριστικό προς εξέταση το οποίο στην πραγματικότητα να ανήκει στην κόκκινη κλάση. Οπτικά φαίνεται ότι βρίσκεται πιο κοντά σε στιγμιότυπο της κλάσης του, αλλά εν τέλει καταλήγει να κατηγοριοποιηθεί στην μπλε κλάση διότι η πλειοψηφία των γειτόνων του ανήκουν σε αυτήν την κλάση. Τέτοιου τύπου προβλήματα στον αλγόριθμο Weighted KNN αντιμετωπίζονται με την χρήση βαρών. Η λογική είναι ότι τα στιγμιότυπα τα οποία βρίσκονται πιο κοντά στο στιγμιότυπο προς εξέταση θα έχουν μεγαλύτερο βάρος από αυτό που βρίσκονται πιο μακριά. Η παραγωγή των βαρών αυτών γίνεται με την χρήση μιας συνάρτησης πυρήνα η οποία θα πρέπει να μικραίνει όσο μεγαλώνει η απόσταση. Για αυτόν το λόγο όλες οι συναρτήσεις οι οποίες πληρούν αυτήν την ιδιότητα μπορούν να χρησιμοποιηθούν. Η κλασσική συνάρτηση η οποία χρησιμοποιείται είναι η αντίστροφη της απόστασης βαρών (IDW) συνάρτηση. Ιδιαίτερα σημαντικό να αναφερθεί είναι πως ο αλγόριθμος αυτός έχει σαν χαρακτηριστικό ότι αυξάνει την πολυπλοκότητα με τον υπολογισμό βαρών και είναι πιο αργός στην εκτέλεση.



Σχήμα 2.8: Η απόσταση Hamming γραφικά.



Σχήμα 2.9: Πρόβλημα που επιλύει ο Weighted KNN.

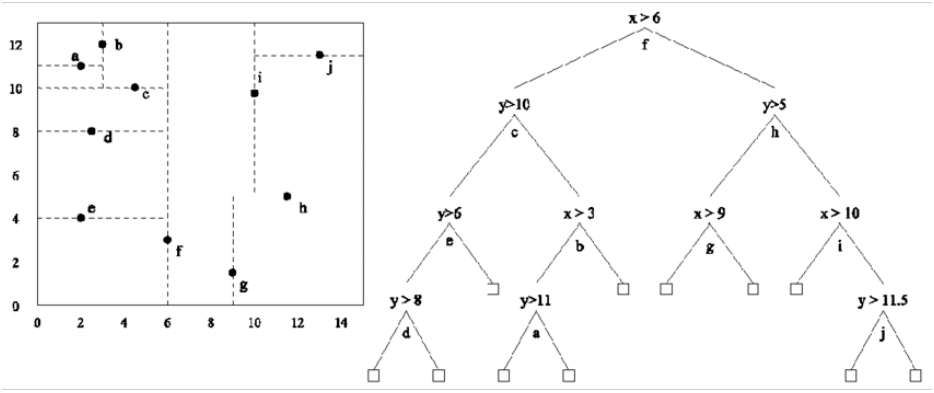
2.5.2 KD Tree KNN

Η αμέσως επόμενη παραλλαγή του αλγορίθμου KNN που θα αναλυθεί είναι ο KD Tree KNN. Ο αλγόριθμος KD Tree KNN είναι μια άλλη παραλλαγή του αλγορίθμου KNN η οποία όμως βασίζεται στα δένδρα κ διαστάσεων (K-Dimensional) [14]. Ένα KD Tree είναι μια δομή δεδομένων η οποία χρησιμοποιείται για την κατάτμηση ενός χώρου σε περιοχές ώστε να ομαδοποιήσει τα σημεία σε ένα χώρο ο οποίος αποτελείται από κ διαστάσεις. Τα δένδρα στα οποία αναφερόμαστε είναι δυαδικά δένδρα στα οποία κάθε κόμβος τους είναι ένα σημείο. Η λογική των KD tree είναι η παρακάτω. Κάθε κόμβος ο οποίος δεν είναι φύλλο, αποτελεί ένα διαχωριστικό του χώρου σε δύο μέρη. Τα σημεία τα οποία βρίσκονται στην αριστερή περιοχή αποτελούν το ένα υπό-δένδρο και τα σημεία τα οποία βρίσκονται στην δεξιά περιοχή αποτελούν το άλλο υπό-δένδρο. Το δένδρο αυτό κατασκευάζεται διαιρώντας τον χώρο των πολλών επιπέδων σε επιμέρους επίπεδα ως προς κάθε διάσταση. Το πρώτο σημείο στο οποίο θα γίνει η αναφορά είναι το πως φτιάχνεται ένα KD δένδρο. Αρχικά ξεκινάμε από ολόκληρο το σύνολο των δεδομένων στον χώρο των κ διαστάσεων. Αρχικά θα γίνει επιλογή μιας από τις διαστάσεις του

στιγμιότυπου. Στην συνέχεια θα γίνει υπολογισμός της διαμέσου από την διάσταση η οποία έχει επιλεχθεί. Ο υπολογισμός της διαμέσου θα γίνει λαμβάνοντας υπόψη όλες τις συντεταγμένες όλων των σημείων από το σύνολο δεδομένων. Σε αυτό το σημείο θα γίνει χωρισμός αυτού του συνόλου δεδομένων σε δύο υποσύνολα. Το πρώτο υποσύνολο θα περιλαμβάνει όλα τα σημεία στιγμιότυπα που έχουν μικρότερη τιμή σε αυτήν την συντεταγμένη και το δεύτερο υποσύνολο θα περιλαμβάνει όλα τα σημεία στιγμιότυπα που έχουν μεγαλύτερη τιμή. Στην συνέχεια αυτή η τιμή η οποία έχει υπολογιστεί θα χρησιμοποιηθεί για να φτιαχτεί ένας κόμβος που θα έχει αυτήν την διάμεση τιμή. Αυτή η διαδικασία θα γίνει αναδρομικά για να χτιστεί το αριστερό και το δεξί υπό-δένδρο. Άρα συνοπτικά γίνεται πρώτα επιλογή μιας διάστασης με την οποία θα χωρίσουμε τα χαρακτηριστικά. Μετά με βάση αυτή την διάσταση βρίσκουμε τον διάμεσο των στιγμιοτύπων για αυτό το χαρακτηριστικό. Και αυτή η διαδικασία επαναλαμβάνεται για όσο αυξάνεται το βάθος του δένδρου. Σε αυτό το σημείο να αναφέρουμε πως υπάρχουν πολλές στρατηγικές επιλογής διάστασης καθώς αυξάνεται το βάθος του δένδρου. Η μια απλή προσέγγιση που μπορούμε να ακολουθήσουμε είναι να γίνεται εναλλάξ επιλογή διάστασης. Σε αυτήν την τακτική στην ρίζα μπορεί να γίνει διαχωρισμός με τον άξονα χ και στα πεδία του κόμβου ρίζα να γίνει με τον άξονα y. Αντίστοιχα μετά εφόσον εκτελείται αναδρομικά η διαδικασία στα καινούργια παιδιά που δημιουργήθηκαν θα γίνει επιλογή της διάστασης x. Αυτή η διαδικασία μπορεί να γίνει μέχρι να δημιουργηθεί το δένδρο. Η δεύτερη στρατηγική επιλογής διάστασης έχει να κάνει με τη διαίρεση του εύρους των συντεταγμένων. Σε αυτήν την περίπτωση ο τρόπος επιλογής της διάστασης που θα χωρίσει το σύνολο των δεδομένων θα γίνει με κριτήριο την διάσταση που έχει το μεγαλύτερο εύρος. Με αυτόν τον τρόπο μπορεί να εξασφαλιστεί ότι δεν επιλέγεται συνέχεια μια συγκεκριμένη διάσταση. Η τρίτη στρατηγική που θα αναφερθεί έχει να κάνει με τον χωρισμό ανάλογα με τις διάμεσους. Σε αυτόν τον τρόπο γίνεται υπολογισμός των διαμέσων για κάθε πιθανή διάσταση και επιλέγεται αυτή η οποία είναι πιο κοντά στο κέντρο. Η τέταρτη και τελευταία στρατηγική που θα αναφερθεί είναι η τυχαία επιλογή διάστασης στην οποία σε κάθε επίπεδο του δένδρου γίνεται με τυχαίο τρόπο χωρισμός του σύνολο δεδομένων.

Η εξαγωγή συμπερασμάτων για τους Κ εγγύτερους γίνεται ως εξής: Ορίζουμε σαν σημείο αρχή την ρίζα του δένδρου και στην συνέχεια γίνεται αναδρομική αναζήτηση στο δένδρο. Η διάτρεξη του δένδρου γίνεται από το αριστερό ή δεξί υπό-δένδρο και έχει μεγαλύτερη τιμή από τον τρέχοντα κόμβο για την διάσταση που εξετάζουμε. Αυτή η διαδικασία γίνεται μέχρι να φτάσουμε σε κόμβο φύλλο ο οποίος θα μπει σε μια λίστα που κρατάμε ταυτόχρονα με τους κ εγγύτερους γείτονες που έχουν βρεθεί μέχρι τώρα. Για όσο η λίστα μας αυτή δεν είναι γεμάτη, δηλαδή το μέγεθος της δεν έχει φτάσει την τιμή κ που είναι ο μέγιστος αριθμός εγγύτερων γειτόνων θα γίνεται πρόσθεση των κόμβων. Εφόσον έχει φτάσει σε κόμβο φύλλο αυτός θεωρείται ο τρέχων καλύτερος, οπότε έξω από την απόσταση που έχει αυτός ο κόμβος σε σχέση με τον κόμβο προς εξέταση δεν πρόκειται να υπάρχει άλλος καλύτερος γείτονας. Αυτό που θα πρέπει να ελεγχθεί είναι εάν υπάρχουν άλλοι κόμβοι οι οποίοι είναι εντός αυτής της ακτίνας. Συνεπώς στην συνέχει θα ξεκινήσει η προς τα πάνω προσπέλαση θα προσπαθήσει να βρει και άλλους κόμβους φύλλα που έχουν μικρότερη απόσταση για να τους προσθέσει στην λίστα των εγγυτέρων γειτόνων. Στο παρακάτω σχήμα μπορούμε να δούμε ένα υπάρχον παράδειγμα ενός KD Tree το οποίο έχει δημιουργηθεί από την εναλλάξ επιλογή διαστάσεων και κατάτμηση τους. Στο δεξί σχήμα βλέπουμε πως έχει δημιουργηθεί ένα δένδρο όπου αρχικά βρίσκει την διάμεση τιμή για τον άξονα x και χωρίζει το σύνολο δεδομένων σε 2 υποσύνολα. Στο αμέσως επόμενο

επίπεδο γίνεται χωρισμός με βάση την διάσταση y και χωρίζει τα 2 υποσύνολα σε 4 υποσύνολα. Αυτή διαδικασία συνεχίζεται μέχρις ότου να φτιαχτεί το δένδρο. Στο αριστερό σχήμα βλέπουμε μια δισδιάστατη απεικόνιση των δεδομένων στον χώρο. Μπορούμε πολύ εύκολα να παρατηρήσουμε πως αρχικά χωρίζοντας τον άξονα x έχει βρει σαν διάμεσο την τιμή 6 και χωρίζει το επίπεδο σε 2 υποεπίπεδα. Αυτό το οποίο πρέπει να σημειωθεί είναι πως κάθε κόμβος που δεν είναι φύλλο από το δεξί σχήμα αποτελεί μια διαχωριστική γραμμή που χωρίζει τα δεδομένα στο αριστερό σχήμα.

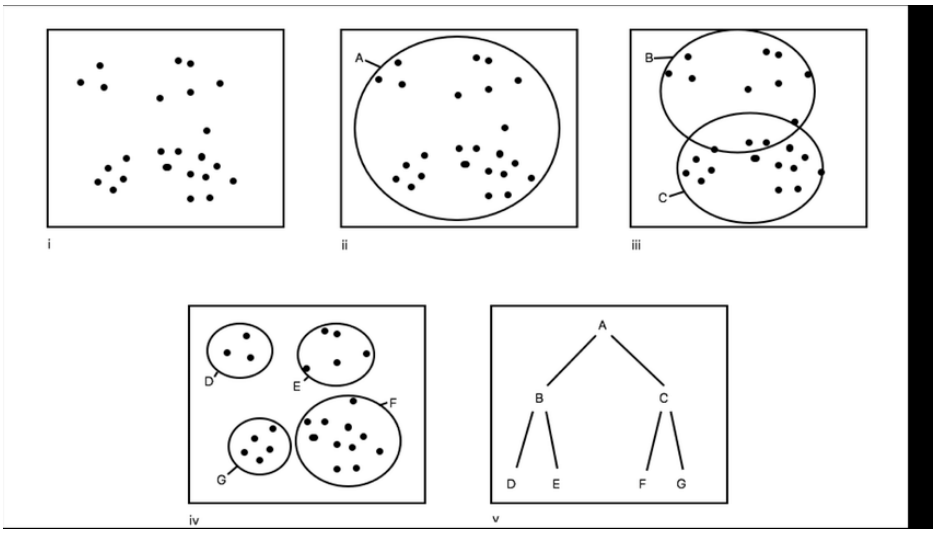


Σχήμα 2.10: Δένδρο KD Tree [15].

2.5.3 Ball Tree KNN

Ο επόμενος αλγόριθμος ο οποίος θα αναλυθεί είναι ο Ball Tree. Ο αλγόριθμος αυτός είναι παραπλήσιος με τον KD-tree με την διαφορά ότι ο αλγόριθμος αυτός δεν παράγει κουτιά όπως φάνηκε στο σχήμα παραπάνω, αλλά δημιουργεί κάποιες μπάλες οι οποίες θα μεταφραστούν με την σειρά τους σε ένα δένδρο. Έστω ότι έχουμε ένα σύνολο δεδομένων με κάποια σημεία σε ένα δισδιάστατο χώρο. Στην αρχή επιλέγουμε ένα σημείο τυχαία έστω T, στην συνέχεια θα βρει εκείνο το σημείο M1 το οποίο είναι πιο μακριά από το επιλεγμένο σημείο. Στην συνέχεια από αυτό το σημείο M1 θα πάμε να βρούμε το πιο απομακρυσμένο του σημείο M2, δηλαδή το σημείο το οποίο έχει την μεγαλύτερη Ευκλείδεια απόσταση. Στην συνέχεια ενώνουμε αυτά τα 2 σημεία M1 και M2 τα οποία ουσιαστικά χωρίζουν το σύνολο δεδομένων στην μέση. Στην συνέχεια γίνεται μια απεικόνιση όλων των δεδομένων του συνόλου δεδομένων πάνω σε αυτό το διάνυσμα. Στην συνέχεια αυτό που πρέπει να γίνει είναι να βρεθεί η διάμεσος αυτού του διανύσματος. Στην συνέχεια όπως έχει χωριστεί αυτό το διάνυσμα από την διάμεσο βρίσκουμε τα μέσα K1, K2 μεταξύ των δυο σημείων M1 και M2 με το σημείο της διαμέσου. Στην συνέχεια από αυτά τα 2 σημεία βρίσκουμε τα πιο μακρινά σημεία τους και οι αποστάσεις αυτές αποτελούν τις ακτίνες των 2 κύκλων που θα παραχθούν. Αυτή η διαδικασία γίνεται αναδρομικά σπάζοντας τους κύκλους σε μικρότερους κύκλους. Βλέπουμε και στο παρακάτω σχήμα 2.11 πως στην φάση i τα δεδομένα δεν ανήκουν σε καμία σφαίρα, στην συνέχειά εφόσον βρεθούν τα πιο μακρινά σημεία δημιουργείται η πρώτη μεγάλη σφαίρα η οποία περιλαμβάνει όλα σημεία του συνόλου δεδομένων. Στην συνέχεια βλέπουμε πως εφόσον έχει βρεθεί το μέσο σημείο μεταξύ των μακρινότερων σημείων έχουν βρεθεί τα κέντρα των 2 κύκλων και έχουν σχηματιστεί οι δύο κύκλοι. Η ίδια διαδικασία γίνεται στην συνέχεια για κάθε κύκλο έως ότου να

καταλήξουμε να έχουμε τους κύκλους D,E,G,F. Τέλος όπως στην διαδικασία των KD Trees δημιουργούμε το δυαδικό δένδρο.
Σε αυτό το σημείο εφόσον έχει δημιουργηθεί το δυαδικό δένδρο, θα πάμε να εφαρμόσουμε τον αλγόριθμο KNN στο δένδρο αυτό. Η διαδικασία ξεκινάει με ένα σημείο για το οποίο εξετάζουμε και ξεκινάμε από την ρίζα του δένδρου. Στην συνέχεια γίνεται υπολογισμός της απόστασης του εξεταζόμενου σημείου με το κέντρο της τρέχουσας σφαίρας, δηλαδή του κόμβου. Αμέσως μετά διατρέχουμε το δένδρο επιλέγοντάς το δένδρο του οποίου η σφαίρα έχει το κέντρο της πιο κοντά στο εξεταζόμενο σημείο. Η διαδικασία αυτή εκτελείται αναδρομικά μέχρι και τα φύλλα του δένδρου.
Σε αυτό το σημείο είναι σημαντικό να κάνουμε μια σύγκριση του αλγορίθμου Ball Tree KNN και του KD Tree KNN. Γενικά είναι άξιο αναφοράς ότι ο αλγόριθμος Ball Tree KNN έχει την ίδια χρήση με τον αλγόριθμο KD Tree KNN, αλλά έχουν διαφορά στην ταχύτητα εκτέλεσης. Γενικά στον Ball-Tree έχουμε μεγαλύτερο χρόνο εκτέλεσης στις λίγες διαστάσεις (λιγότερες από 4 διαστάσεις) σε αντίθεση με τις πολλές διαστάσεις που ο αλγόριθμος Ball Tree είναι πολύ πιο γρήγορος. Μια άλλη διαφορά είναι ότι ο αλγόριθμος Ball Tree μπορεί να δουλεύει καλά και σε πολλές περισσότερες διαστάσεις όταν έχει κάποιες τοπικές δομές.



Σχήμα 2.11: Η διαδικασία κατάτμηση του συνόλου δεδομένων σε σφαίρες [16]

Μια άλλη παραλλαγή του αλγορίθμου KNN είναι ο KNN με τον K-Means. Ένα χαρακτηριστικό του KNN είναι ότι έχει πολύ μεγάλη υπολογιστική πολυπλοκότητα. Όπως φαίνεται έχει προταθεί μια παραλλαγή η οποία προσπαθεί να αντιμετωπίσει αυτό το μειονέκτημα του αλγορίθμου χρησιμοποιώντας την τεχνική που χρησιμοποιείται στον αλγόριθμο K-Means που χρησιμοποιείται για συσταδοποίηση. Στο αλγόριθμο που παρουσιάζεται πρώτα θα γίνει υπολογισμός των συστάδων για τις διάφορες κατηγορίες των δεδομένων εκπαίδευσης. Κάθε μια συστάδα έχει ένα κέντρο και όλα αυτά τα κέντρα θα χρησιμοποιηθούν σαν νέα δεδομένα εκπαίδευσης. Ο αλγόριθμος αυτός θα χρησιμοποιηθεί για να εξάγει αποτέλεσμα με τον παρακάτω τρόπο: Αρχικά θα χρησιμοποιήσει τα κέντρα αυτά για να υπολογίσει τις αποστάσεις από το νέο στιγμιότυπο προς εξέταση. Στην συνέχεια αυτό το στιγμιότυπο θα ανήκει στη κατηγορία με την οποία έχει την μικρότερη απόσταση. Σημαντικό να αναφερθεί είναι ότι σε κανένα σημείο της διαδικασίας δεν αναφέρθηκε η μεταβλητή k η οποία χρησιμοποιείται στον κλασσικό αλγόριθμο KNN

καθώς το νέο στιγμιότυπο θα ανήκει απλά στο κοντινότερο κέντρο κάθε κατηγορίας.

2.5.4 Bayesian-kNN

Μια άλλη παραλλαγή του KNN η οποία θα ήταν χρήσιμο να αναφερθεί είναι ο Bayesian KNN [17]. Είναι σημαντικό να αναφερθεί πως το αλγόριθμος Bayes δεν δίνει ως αποτέλεσμα την κατηγορία ενός στιγμιότυπου, αλλά αντίθετα δίνει την συμμετοχή σε κάθε κατηγορία. Η νέα παραλλαγή του Bayes και του K-NN με στόχο την αύξηση της απόδοσης του KNN δημιουργήθηκε για την βελτίωση αυτής της κατάταξης. Ο αλγόριθμος αυτός είναι μια επέκταση του κλασσικού KNN με βάση τις πιθανότητες. Στον κλασσικό KNN γίνεται εξαγωγή συμπεράσματος για την κατηγορία του στιγμιότυπου προς εξέταση με βάση την πλειοψηφία της κλάσης των γειτόνων του. Το καινούργιο χαρακτηριστικό το οποίο προστίθεται από την μεριά του Bayes είναι η δημιουργία κατανομών πιθανότητας αλλά και αβεβαιότητας για κάθε κλάση. Η πρώτη φάση που υπάρχει στον Bayesian KNN είναι αυτή της εκπαίδευσης όπου γίνεται η μορφοποίηση των δεδομένων σε μορφή διανυσμάτων με τα χαρακτηριστικά και την κατηγορία τους. Στην συνέχεια για κάθε στιγμιότυπο θα γίνει μια εκτίμηση μιας κατανομής, όπως η κατανομή Gauss, όπως επίσης και υπολογισμός της εκ των προτέρων πιθανότητας για το κατά πόσο ανήκει κάθε στιγμιότυπο σε μια κατηγορία. Αμέσως μετά ακολουθεί η φάση της εκτίμηση όπου έρχεται ένα στιγμιότυπο προς εξέταση και εκτελείται κανονικά ο αλγόριθμος των κ εγγύτερων γειτόνων, στην συνέχεια όμως δεν γίνεται ανάθεση αυτού του κ σε αυτό το στιγμιότυπο με την πλειοψηφούσα κατηγορία αλλά ο BKNN δημιουργεί της κατανομές πιθανότητας για κάθε έναν γείτονα. Άρα γίνεται υπολογισμός της πιθανότητας να ανήκει ένα στιγμιότυπο σε κάθε κατηγορία. Στην συνέχεια η κατηγορία με την μεγαλύτερη εκ των υστέρων πιθανότητα ορίζεται ως η κλάση του στιγμιότυπου.

2.5.5 DB-kNN

Ο επόμενος αλγόριθμος που θα αναφερθεί είναι ο DB-kNN δηλαδή ο Density Based kNN [18]. Αυτός ο αλγόριθμος αποτελεί μια επέκταση του αλγορίθμου KNN με σκοπό να λαμβάνονται και άλλοι παράγοντες υπόψη. Ένας τέτοιος παράγοντας είναι η πυκνότητά του συνόλου δεδομένων. Σε αυτήν την επέκταση του αλγορίθμου KNN μπορούμε να εντοπίσουμε μια καινούργια έννοια η οποία είναι η δομική πυκνότητά (Structural Density). Η δομική πυκνότητά είναι ένας αριθμός, οποίος θα επηρεάσει τον αλγόριθμο. Ο αριθμός αυτός ισούται με τον αριθμό των σημείων που υπάρχουν στην γειτονία ενός σημείου δια τον αριθμό του όγκου της γειτονιάς. Η γειτονιά ενός στοιχείου είναι η περιοχή που βρίσκεται ένα στοιχείο. Συνήθως η γειτονιά θα οριστεί από μια ακτίνα από ένα σημείο. Ο όγκος αυτής της γειτονίας ορίζεται από την γεωμετρία του χώρου που δουλεύουμε. Ένα παράδειγμα είναι αν ήμαστε σε 3D χώρο να είναι μια σφαίρα. Τέλος τα στοιχεία που υπάρχουν μέσα σε αυτόν τον όγκο είναι η πληροφορία για να υπολογιστεί η πυκνότητα των στοιχείων αυτών. Ένα σημείο στο οποίο δεν έχουμε αναφερθεί ακόμα είναι το σε ποιον όγκο θα ψάξουμε να βρούμε την πυκνότητά. Σε αυτό το σημείο υπάρχει μια παράμετρος r (radius) του αλγορίθμου η οποία ορίζει την ακτίνα στην οποία θα ορίσουμε την πυκνότητά. Για αρχή θα γίνει υπολογισμός της πυκνότητας σε αυτόν τον όγκο και στην συνέχεια θα γίνει υπολογισμός της μέσης πυκνότητάς σε όλο το σύνολο δεδομένων. Αυτή η μέση πυκνότητα θα υπολογιστεί από την διαίρεση του συνολικού αριθμού των στοιχείων του συνόλου δεδομένων με τον όγκο που ορίζει το σύνολο δεδομένων. Αυτό το οποίο πρέπει να

γίνει είναι αναζήτηση μιας τιμής της ακτίνας ώστε το μέσο από τις διαφορετικές πυκνότητες να είναι ίσο με την μέση πυκνότητα που υπολογίζεται σε ολόκληρο το σύνολο δεδομένων. Ο στόχος δηλαδή είναι να βρεθεί μια ακτίνα ώστε όλες οι τοπικές πυκνότητες στις γειτονιές τους να ταιριάζουν με την συνολική πυκνότητα του συνόλου δεδομένων. Το κίνητρο το οποίο υπάρχει πίσω από αυτήν την διαδικασία είναι να οριστεί ένα μέγεθος γειτονιάς το οποίο να λαμβάνει υπόψη κάθε πυκνότητα σε κάθε γειτονιά που υπάρχει στο σύνολο δεδομένων. Εν κατακλείδι, η Δομική Πυκνότητα είναι ένα μέτρο που εξετάζει την πυκνότητα των σημείων μέσα σε μια γειτονιά που υπάρχει σε μια ακτίνα από ένα στοιχείο. Η ακτίνα αυτής της γειτονιάς θα προσδιοριστεί αναζητώντας μια τιμή που κάνει τη μέση τοπική πυκνότητα να ταιριάζει με τη συνολική πυκνότητα του συνόλου δεδομένων. Αυτή η διαδικασία έγινε για την αξιολόγηση της σημαντικότητας της κάθε γειτονιάς. Αμέσως μετά από τον υπολογισμό των πυκνοτήτων γίνεται κανονικοποίηση στο διάστημα [0-1] καθώς η σχετικές πυκνότητες φαίνεται να είναι πιο χρήσιμες από τις απόλυτες. Στην συνέχεια βασιζόμενοι σε αυτές τις πυκνότητες κάθε γείτονας αξιολογηται σαν στοιχείο πυρήνας για την κατηγορία που ανήκει υπολογίζοντας την σχετική δομική πυκνότητα για την κλάση που ανήκει.

Κεφάλαιο 3

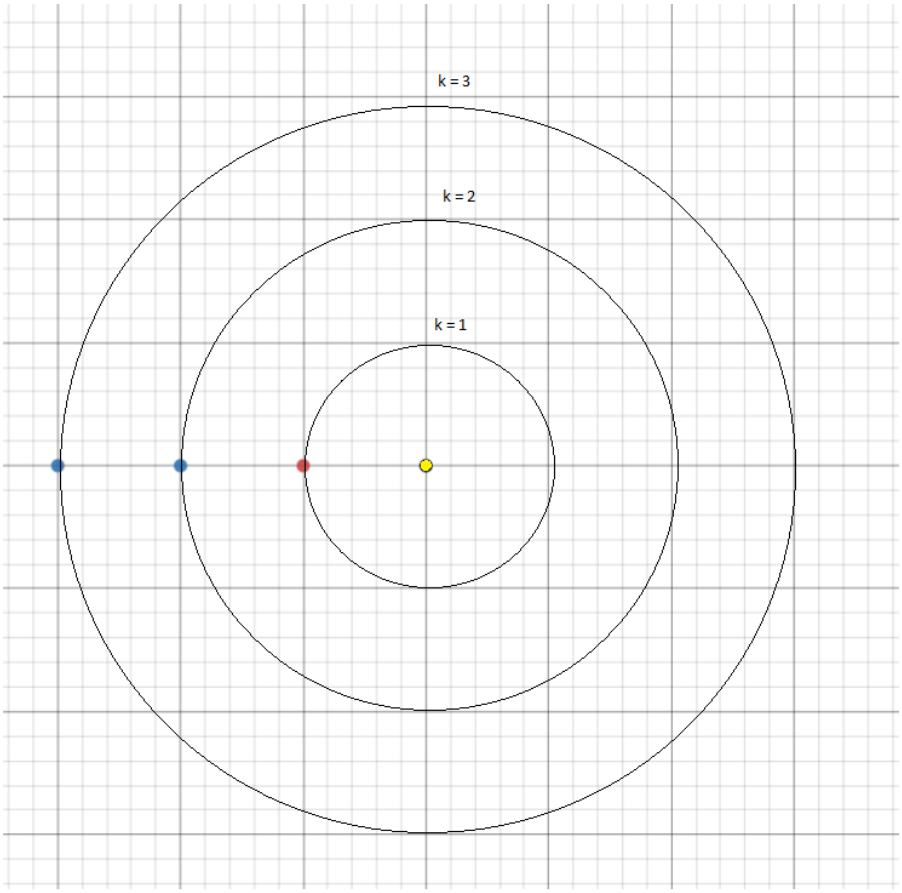
Ο αλγόριθμος AdaNN και προτεινόμενες παραλλαγές

3.1 Ο αλγόριθμος AdaNN

Στο κεφάλαιο αυτό θα δοθεί βάση στον αλγόριθμο AdaNN τον οποίο θα γίνει προσπάθεια να βελτιώσουμε. Όπως φαίνεται ο αλγόριθμος K-NN έχει διάφορα μειονεκτήματα τα οποία πρέπει να αντιμετωπιστούν κυρίως όσον αφορά το σταθερό κ με το οποίο γίνεται η ταξινόμηση των προτύπων. Όπως φαίνεται θεωρείται πως για την σωστή εφαρμογή του αλγορίθμου KNN πρέπει τα στιγμιότυπα τα οποία θα χρησιμοποιηθούν είναι ακριβώς ίσα κατανεμημένα στις διαφορετικές κατηγορίες που έχουμε στο πρόβλημα μας. Στην πραγματικότητα αυτό όμως είναι λογικό να μην συμβαίνει και τα δεδομένα μας να είναι unbalanced. Σε ένα unbalanced σύνολο δεδομένων η πλειοψηφούσα κλάση εκπροσωπείται από πολύ περισσότερα στιγμιότυπα από τις άλλες κλάσεις οι οποίες εκπροσωπούνται από πιο χαμηλά ποσοστά. Στην διαδικασία που ακολουθείται από τον αλγόριθμο KNN γίνεται ανίχνευση της γειτονιάς στην οποία βρίσκεται το στιγμιότυπο προς εξέταση και στην συνέχεια ανάλογα με την τοπική πλειοψηφούσα κλάση θα ταξινομηθεί ανάλογα το στιγμιότυπο. Λαμβάνοντας ως δεδομένο ότι μια κατηγορία έχει γενικότερα περισσότερα στιγμιότυπα η σωστή επιλογή του κ είναι μια πολύ κρίσιμη διαδικασία για την εξαγωγή συμπεράσματος. Είναι μάλιστα πιθανό να υπάρχει μια προκατάληψη προς την πλειοψηφούσα κλάση. Η πρόταση του αλγορίθμου AdaNN αποτελεί μια βελτιωμένη έκδοση του αλγορίθμου κ εγγύτερων γειτόνων. Δεδομένου ότι στιγμιότυπα τα οποία είναι πιο κοντινά έχουνε και χαρακτηριστικά με παρόμοιες τιμές τότε είναι πιθανό αυτά τα δύο χαρακτηριστικά με την υψηλή ομοιότητα και μικρή απόσταση να κατηγοριοποιούνται σωστά με τον ίδιο αλγόριθμο κ εγγύτερων γειτόνων. Σε αυτό το σημείο πρέπει να αναφερθούμε και στην έννοια του βέλτιστου κ. Αυτός ο αριθμός είναι ο μικρότερος αριθμός εγγύτερων γειτόνων για τον οποίο θα μπορεί να κατηγοριοποιηθεί σωστά ένα στιγμιότυπο.
Σε αυτό το σημείο είναι ιδιαίτερα σημαντικό να γίνει αναφορά στα βήματα τα οποία θα χρειαστούν για την εκτέλεση του αλγορίθμου AdaNN. Το βασικό σημείο στο οποίο γίνεται η διαφοροποίηση με τον αλγόριθμο KNN είναι ότι κάθε διαφορετικό στιγμιότυπο έχει ένα διαφορετικό αριθμό κ για τον οποίον θα κατηγοριοποιηθεί το στιγμιότυπο. Συνεπώς κάπως θα πρέπει να υπολογιστεί η τιμή αυτού του βέλτιστου κ. Για αρχή αυτό που γίνεται είναι η εκτέλεση ενός αλγορίθμου 9 NN. Για αρχή θα πρέπει να γίνει κατανοητό το τι εισόδους και τι εξόδους παίρνει ο αλγόριθμος. Σαν είσοδο θα πάρει τα δεδομένα της εκπαίδευσης από το σύνολο δεδομένων

μας. Από την άλλη έξοδος του αλγορίθμου θα είναι το σύνολο των βέλτιστων κ για κάθε στοιχείο του συνόλου της εκπαίδευσης μας. Σε αυτό το σημείο θα πρέπει να γίνει αντιληπτός ο ορισμός του βέλτιστου κ ο οποίος σημαίνει την μικρότερη τιμή κ με την οποία θα εκτελεστεί ο αλγόριθμος κ εγγύτερων γειτόνων και ταυτόχρονα αυτό το στιγμιότυπο να κατηγοριοποιηθεί στην σωστή κλάση. Ουσιαστικά αυτό που αναρωτιόμαστε είναι "πόσους γείτονες χρειάζεται να ρωτήσουμε ώστε να κατηγοριοποιήσουμε σωστά ένα στιγμιότυπο;". Για αυτόν τον λόγο θα πρέπει να γίνει μια προσπέλαση όλων των κ από την τιμή 1 έως την τιμή 9. Η τακτική η οποία θα ακολουθηθεί είναι να γίνει διαδοχική εκτέλεση του αλγορίθμου KNN από τιμή 1 έως την τιμή 9. Στην περίπτωση που το στιγμιότυπο κατηγοριοποιηθεί σωστά, τότε αυτό το στιγμιότυπο θα έχει ως βέλτιστο κ αυτό με το οποίο θα κατηγοριοποιηθεί σωστά. Σε περίπτωση που φτάσουμε την μέγιστη τιμή του 9 τότε θα οριστεί ως βέλτιστη τιμή κ για εκείνο το στιγμιότυπο η τιμή 9. Να σημειωθεί σε αυτό το σημείο ότι η έννοια της εγγύτητας σε αυτόν τον αλγόριθμο θα υπολογιστεί με την χρήση της μετρικής της Ευκλείδειας απόστασης μεταξύ του τρέχοντος στιγμιότυπου και των υπόλοιπων δεδομένων της εκπαίδευσης. Εφόσον γίνει υπολογισμός όλων των αποστάσεων θα γίνει ταξινόμηση των αποστάσεων αυτών και θα γίνει διαλογή των 9 στιγμιοτύπων με τις κοντινότερες αποστάσεις από το εξεταζόμενο. Αμέσως μετά είναι που θα γίνει η εκτέλεση του αλγορίθμου AdaNN. Βλέπουμε δηλαδή πως χρειάζεται μια προεργασία πριν την εφαρμογή του. Ο αλγόριθμος αυτός θα πάρει σαν εισόδους το σύνολο των δεδομένων εκπαίδευσης καθώς και το βέλτιστο κ για κάθε ένα από αυτά τα στιγμιότυπα. Η τελευταία παράμετρος που θα χρειαστεί ο αλγόριθμος είναι το σύνολο των δεδομένων ελέγχου τα οποία θα τεθούν προς ταξινόμηση προκειμένου να αξιολογηθεί η ικανότητα γενίκευσης του αλγορίθμου. Τέλος η έξοδος του αλγορίθμου θα είναι η τιμή της ακρίβειας (accuracy) που θα πετύχει ο αλγόριθμος. Η διαδικασία η οποία θα ακολουθηθεί είναι η παρακάτω: Για αρχή θα γίνει μια προσπέλαση σε όλα τα στιγμιότυπα του συνόλου ελέγχου. Για κάθε ένα στιγμιότυπο από το σύνολο ελέγχου θα γίνει εκτέλεση του αλγορίθμου εγγύτερων γειτόνων έτσι ώστε από το στιγμιότυπου του συνόλου ελέγχου να εντοπιστεί το στιγμιότυπο του συνόλου εκπαίδευσης με το οποίο έχει την μικρότερη απόσταση με το τρέχον. Αμέσως μετά από την εύρεση του στιγμιότυπου από το σύνολο εκπαίδευσης θα γίνει λήψη του βέλτιστου κ το οποίο έχει υπολογιστεί στο προηγούμενο βήμα και θα το υιοθετήσει για δικό του. Ουσιαστικά με αυτόν τον τρόπο εφαρμόζεται η τακτική ότι τα κοντινότερα στιγμιότυπα κατηγοριοποιούνται με τον ίδιο αλγόριθμο KNN. Εφόσον το κοντινότερο του στιγμιότυπο από το σύνολο εκπαίδευσης κατηγοριοποιείται με έναν κ αριθμό γειτόνων, με τον ίδιο αριθμό κ θεωρητικά θα κατηγοριοποιηθεί σωστά το εξεταζόμενο στιγμιότυπο. Συνεπώς στην συνέχεια με αυτό το βέλτιστο κ που έχει βρεθεί θα γίνει εκτέλεση αλγορίθμου KNN για το εξεταζόμενο στιγμιότυπο ελέγχου έτσι ώστε να βρεθεί η πλειοψηφούσα κατηγορία στην γειτονιά του και να βρει την τελική του κατηγορία με βάση την πλειοψηφία της γειτονιάς του. Αμέσως μετά θα γίνει υπολογισμός του αριθμού των στιγμιοτύπων τα οποία εν τέλει κατηγοριοποιήθηκαν σωστά. Αυτός ο αριθμός θα μας χρησιμεύσει ώστε να βρεθεί η ακρίβεια του αλγορίθμου. Σε αυτό το σημείο εφόσον έχουμε τον αριθμό των σωστά ταξινομημένων στιγμιοτύπων από τον σύνολο ελέγχου, είναι η στιγμή να υπολογιστεί η ακρίβεια του αλγορίθμου διαιρώντας το σύνολο των σωστά ταξινομημένων στιγμιοτύπων ελέγχου προς τον συνολικό τους αριθμό.

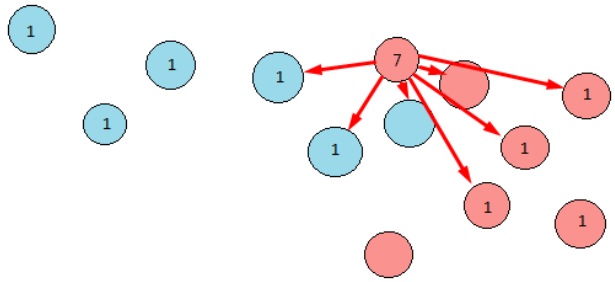
Σημαντικό να αναφερθεί είναι ότι στην εκτέλεση του αλγορίθμου AdaNN όταν θα γίνει υπολογισμός αυτός θα πρέπει κάθε στιγμιότυπο για το οποίο υπολογίζεται το κ να υπολογιστούν οι αποστάσεις των κοντινότερων στιγμιοτύπων που υπάρχουν ώστε να γίνει σωστή ταξινόμηση του στιγμιότυπου.



Σχήμα 3.1: Παράδειγμα εύρεσης του βέλτιστου κ

3.2 Μειονεκτήματα του αλγορίθμου AdaNN

Σε αυτό το σημείο πρέπει να γίνει αναφορά στα διάφορα μειονεκτήματα τα οποία περιλαμβάνει αυτή η διαδικασία της εκτέλεσης του AdaNN. Πιο συγκεκριμένα βλέπουμε πως θέλει μια προεργασία υπολογισμού του βέλτιστου κ για κάθε στιγμιότυπο του συνόλου εκπαίδευσης. Συνεπώς θα χρειαστούν 2 φάσεις. Μια η οποία να κάνει την προεργασία του υπολογισμού των κ, με την έμμεση κατασκευή ενός μοντέλου με βάση το οποίο θα ταξινομηθούν τα στιγμιότυπα του συνόλου ελέγχου. Συνεπώς και πλέον ακολουθούνται τα διάφορα μειονεκτήματα τα οποία έχουν οι οκνηροί κατηγοριοποιητές. Τέτοιο μειονέκτημά όπως αναλύθηκε εκτενώς στην ενότητα 2 είναι η διαρκής επανεκπαίδευση του μοντέλου το οποίο έχουμε με νέα δεδομένα. Σε αυτό το σημείο είναι σημαντικό να αναφερθεί πως εφόσον θα έρθουν καινούργια δεδομένα στο σύνολο εκπαίδευσης, θα αλλάξει η διάρθρωση του και συνεπώς θα πρέπει να γίνει επανυπολογισμός όλων των βέλτιστων κ. Επίσης σημαντικό στοιχείο είναι πως όπως αναφέρεται στο άρθρο [5] θα πρέπει να εκτελεστεί αλγόριθμος προεργασίας των δεδομένων εκπαίδευσης προκειμένου να βρεθούν αυτά τα βέλτιστα κ. Έχοντας ως βάση την επιλογή του αλγορίθμου να ορίσει το εύρος του κ από το 1 έως το 9 ρίχνει την επίδοση του αλγορίθμου σε σύνολα δεδομένων τα οποία βρίσκουν μεγαλύτερη τιμή ακρίβειας όταν η τιμή του κ είναι μεγαλύτερη από 9. Ουσιαστικά για όσα δεν μπορούν να κατηγοριοποιηθούν μέχρι την τιμή των 9 κοντινότερων



Σχήμα 3.2: Παράδειγμα εύρεσης του βέλτιστου κ για στιγμιότυπο κοντά στο σημείο διαχωρισμού των κλάσεων

γειτόνων θα θέσουμε από μόνοι μας αυτήν την τιμή ως βέλτιστο κ.

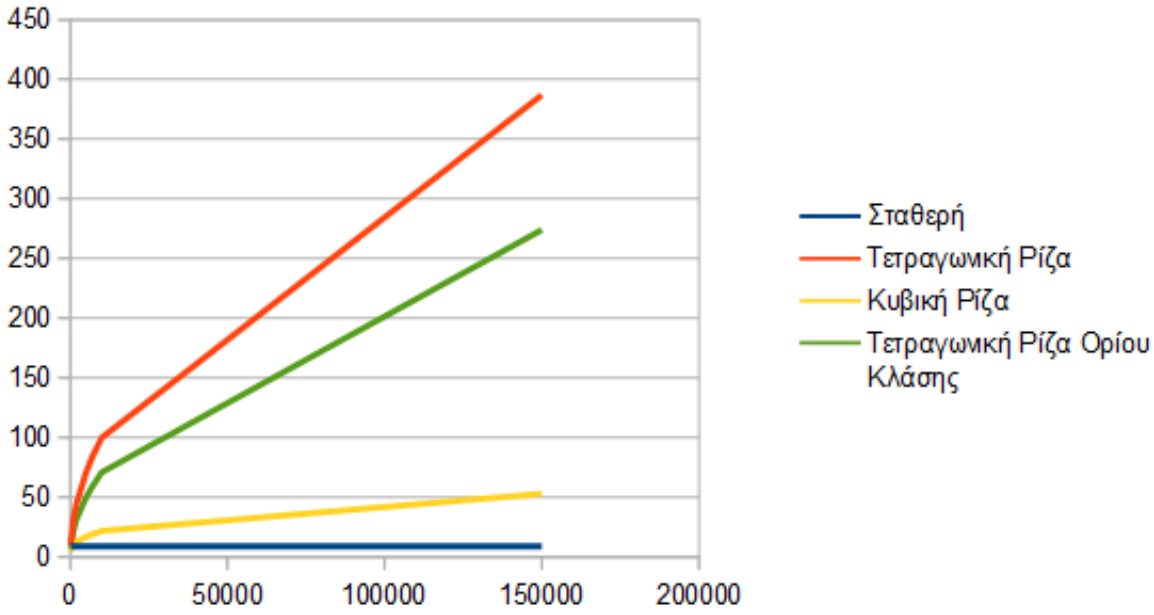
3.3 Απλές παραλλαγές του AdaNN

Σε αυτό το σημείο είναι πολύ σημαντικό να γίνει αναφορά στις παραλλαγές οι οποίες υλοποιήθηκαν στην διπλωματική αυτή. Θα γίνει αναφορά σε ένα πρόβλημα το οποίο έχει να κάνει με την προεργασία της εκτέλεσης του AdaNN και συγκεκριμένα με τις περιπτώσεις όπου ένα στιγμιότυπο δεν θα μπορεί να κατηγοριοποιηθεί με τις τιμές 1 έως 9 όπως ορίζεται στον αλγόριθμο AdaNN. Είναι πολύ λογικό σε πολύ μεγάλα σύνολα δεδομένων ένα στιγμιότυπο να έχει ένα βέλτιστο κ το οποίο να έχει πολύ μεγαλύτερη τιμή από το 9. Στην απλή παραλλαγή του AdaNN, γίνεται διαδοχικός έλεγχος των τιμών από το 1 έως το 9 προκειμένου να βρεθεί το βέλτιστο κ. Σε περιπτώσεις που υπάρχουν όμως μεγάλα σύνολα δεδομένων, αυτό δεν θα είναι εφικτό. Για αυτόν τον λόγο θεωρείται σημαντικό στην παρούσα διπλωματική αυτό το όριο να μπορεί να μεταβληθεί ανάλογα με πλήθος των στιγμιοτύπων του συνόλου δεδομένων ή μιας άλλης τιμής που να έχει σχέση με το πλήθος των στιγμιοτύπων. Σε αυτό το σημείο είναι που μπαίνει ο πρώτος προβληματισμός με την δημιουργία μιας συνάρτησης η οποία θα υπολογίζει το όριο μέχρι ο το οποίο θα ψάξει για το βέλτιστο αυτό κ αντί δηλαδή για την συγκεκριμένη τιμή 9. Γενικά υπήρξε η ανάγκη για μια συνάρτηση η οποία να αυξάνεται όσο αυξάνεται και ο αριθμός των στιγμιοτύπων χωρίς όμως να αυξάνεται πολύ γρήγορα. Οι συναρτήσεις ο οποίες χρησιμοποιήθηκαν είναι η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων, η κυβική ρίζα του αριθμού των στιγμιοτύπων και η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων της κλάσης

του εξεταζόμενου στιγμιότυπου. Αυτός ο αριθμός θα συγκριθεί με την σταθερή συνάρτηση η οποία ισούται με 9 όταν ένα στιγμιότυπο δεν βρίσκει το βέλτιστο του κ έως την τιμή 9. Να σημειωθεί πως για τον υπολογισμό των ορίων του βέλτιστου κ για τον παρακάτω πίνακα στην στήλη της τετραγωνικής ρίζας του ορίου των στιγμιοτύπων της κλάσης λαμβάνουμε υπόψη πως υπάρχουν δύο κατηγορίες και πως το σύνολο των δεδομένων μας είναι τέλεια ισορροπημένο, δηλαδή ότι στην περίπτωση όπου ο αριθμός των στιγμιοτύπων μας είναι 100 τότε θα έχουμε 2 κλάσεις, όπου κάθε κλάση θα έχει από 50 στιγμιότυπα. Συνεπώς το όριο της κλάσης θα είναι η τετραγωνική ρίζα του 50, άρα δηλαδή το αποτέλεσμα θα είναι η τιμή 7.

Αριθμός Στιγμιοτύπων X	Τετραγωνική Ρίζα	Κυβική Ρίζα	Σταθερή	Τετρ. Ρίζα Ορίου Κλάσης
100	10	5	9	7
1000	32	10	9	22
2000	45	13	9	32
5000	71	17	9	50
7000	84	19	9	59
10000	100	22	9	71
15000	387	53	9	274

Πίνακας 3.1: Σύγκριση Συναρτήσεων Υπολογισμού Ορίου κ



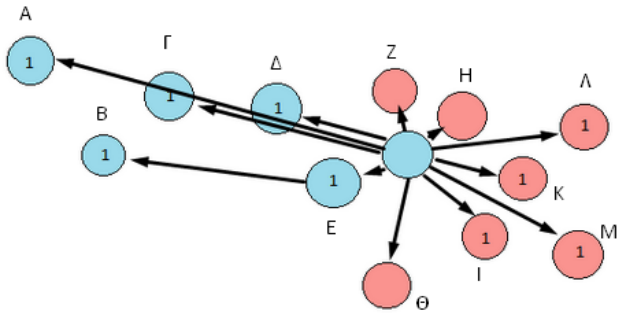
Σχήμα 3.3: Σύγκριση των ρυθμών αύξησης των συναρτήσεων υπολογισμού του βέλτιστου κ

3.4 Παραλλαγές του AdaNN για αφαίρεση του θορύβου

Σε αυτό το σημείο είναι ανάγκη να γίνει αναφορά στην διαχείριση των στιγμιοτύπων τα οποία δεν μπορούν να κατηγοριοποιηθούν και δεν μπορεί να βρεθεί αυτό το βέλτιστο κ. Στα σύνολα των δεδομένων υπάρχουν διάφορα στιγμιότυπα τα οποία δεν μπορούν να ταξινομηθούν με κάποιο κ ακόμα και με την ταυτόχρονη αλλαγή του ορίου του βέλτιστου κ το οποίο είναι εξαρτημένο από το μέγεθος του συνόλου δεδομένων. Για αυτό τον λόγο εφόσον αυτά τα στιγμιότυπα δεν μπορούν να ταξινομηθούν σωστά, τότε θεωρούμε πως πιθανό να είναι θόρυβος. Για αυτόν τον λόγο θα πρέπει να γίνει μια διαχείριση για αυτά τα στιγμιότυπα τα οποία υπάρχουν στο σύνολο δεδομένων και ακόμα και με πολύ μεγάλες τιμές του κ δεν θα κατηγοριοποιηθούν. Το ότι δεν υπάρχει κάποια τέτοια τιμή κ σημαίνει ότι ακόμα και να έρθει ένα καινούργιο υπο εξέταση στιγμιότυπο κοντά σε αυτό το προβληματικό στιγμιότυπο και πάρει από το προβληματικό στιγμιότυπο το κ του είναι πολύ πιθανό και το ίδιο να μην μπορεί να κατηγοριοποιηθεί σωστά. Τέτοιες περιπτώσεις μπορεί να συμβούν αν φτάσουμε στο μέγιστο όριο του κ που έχει οριστεί χωρίς να μπορεί να γίνει η ταξινόμηση. Όπως φαίνεται και στο παρακάτω σχήμα 3.4 υπολογίζονται διαδοχικά οι αποστάσεις με όλα στιγμιότυπα του συνόλου δεδομένων. Παρόλο που έχουμε φτάσει στο τέλος του συνόλου δεδομένων για το εξεταζόμενο στιγμιότυπο που ανήκει στην μπλε κλάση συνεχίζουμε να μην μπορούμε να βρούμε το βέλτιστο κ με το οποίο θα μπορεί να κατηγοριοποιηθεί. Στην περίπτωση που βλέπουμε στο παράδειγμα θεωρούμε πως το στιγμιότυπο αυτό είναι θόρυβος και θα πρέπει να αφαιρεθεί από το σύνολο των δεδομένων. Σημαντικό σημείο είναι πως στο παράδειγμα αυτό εξετάστηκαν όλες οι αποστάσεις για όλα τα στιγμιότυπα του συνόλου δεδομένων, δηλαδή στο μέγιστο όριο στιγμιοτύπων που μπορεί να ρωτήσει ένα στιγμιότυπο προκειμένου να κατηγοριοποιηθεί σωστά. Ουσιαστικά η συνάρτηση από την οποία προέκυψε το όριο εξέτασης των στιγμιοτύπων είναι η ταυτοτική συνάρτηση.

3.4.1 AdaNN NR One Pass

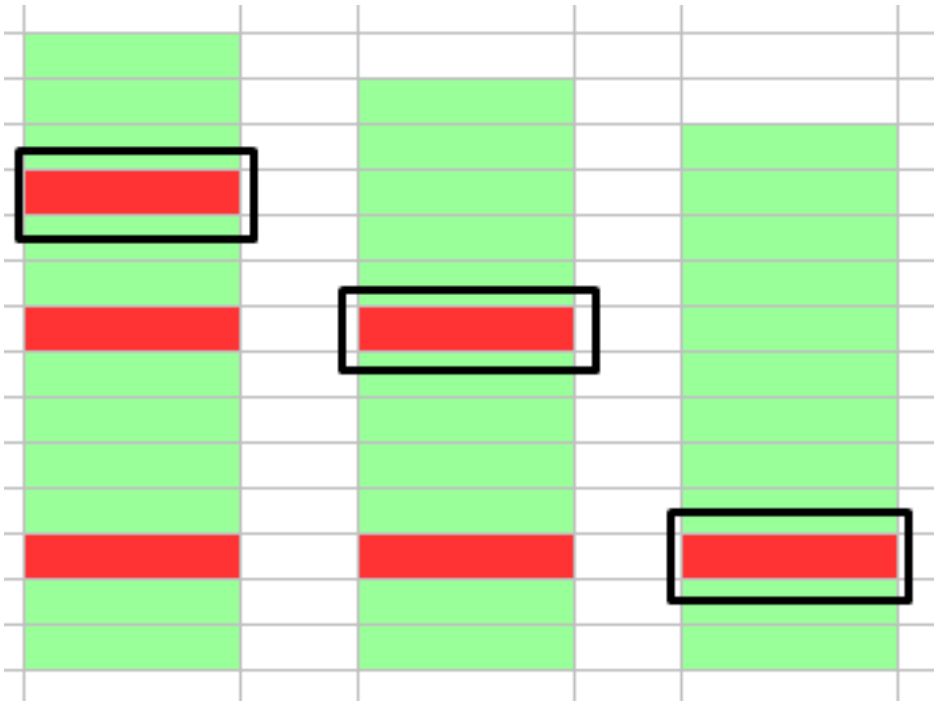
Η πρώτη προτεινόμενη παραλλαγή η οποία θα αναλυθεί και θα αφορά την αφαίρεση του θορύβου είναι η μέθοδος AdaNN NR One Pass. Σε αυτήν την μέθοδο η τακτική η οποία θα ακολουθηθεί είναι κατά την προ-επεξεργασία του συνόλου δεδομένων. Όταν θα πάει να γίνει υπολογισμός του βέλτιστου κ για κάθε στιγμιότυπο του συνόλου δεδομένων, σε περίπτωση που φτάσει στο όριο το οποίο έχουμε θέσει και δεν μπορεί να κατηγοριοποιηθεί με κάποιο ανεκτό όριο του βέλτιστου κ τότε θα ορίσουμε σαν βέλτιστο κ την τιμή -1. Δηλαδή ότι αυτό το στιγμιότυπο είναι θόρυβος και ότι δεν μπορεί να κατηγοριοποιηθεί σωστά. Στην συνέχεια ήδη από την φάση της προ-επεξεργασίας όταν θα γίνει ο πρώτος υπολογισμός του βέλτιστου κ για κάθε στιγμιότυπο του συνόλου δεδομένων, θα γίνει μια διαλογή αυτών των στιγμιοτύπων που έχουν ως βέλτιστο κ την τιμή -1, δηλαδή αυτών εγγραφών που είναι θόρυβος. Οπότε αμέσως μετά αν N είναι ο αριθμός των στιγμιοτύπων και υπάρχουν M στιγμιότυπα τα οποία είναι θόρυβος, ο αλγόριθμος κατά την φάση της προ-επεξεργασίας θα υπολογίσει N αριθμούς για βέλτιστα κ και θα κάνει ένα πέρασμα από τα στιγμιότυπα στην συνέχεια για να αφαιρεθούν τα στιγμιότυπα τα οποία έχουν την τιμή του -1. Με αυτόν τον τρόπο στην συνέχεια όταν θα πάει να γίνει ταξινόμηση των στιγμιοτύπων ελέγχου για να γίνει υπολογισμός της ακρίβειας του αλγορίθμου, θα γίνει μόνο με βάση τα στιγμιότυπα τα οποία δεν αποτελούν θόρυβο, και συνεπώς περιμένουμε ότι θα υπάρξουν καλύτερα αποτελέσματα.



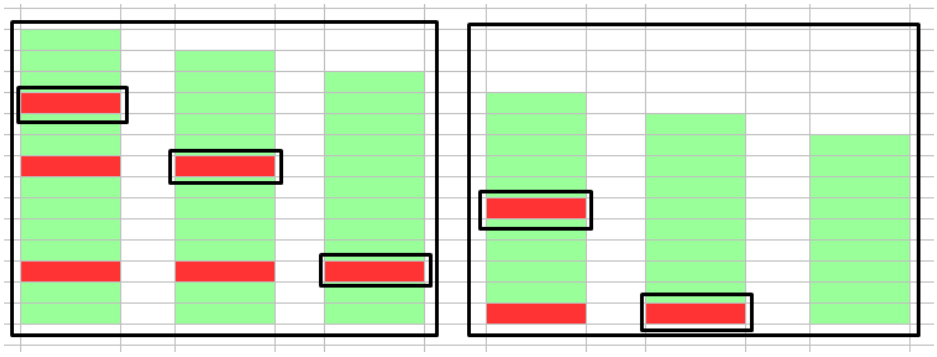
Σχήμα 3.4: Υπολογισμός κ για στιγμιότυπο θόρυβος

3.4.2 AdaNN NR MultiPass

Η επόμενη προτεινόμενη παραλλαγή η οποία θα αναλυθεί έχει να κάνει πάλι με την αφαίρεση του θορύβου στο σύνολο δεδομένων κατά την εκτέλεση του αλγορίθμου. Σε αυτήν την μέθοδο μπορούμε να βρούμε ομοιότητες με την προηγούμενη μέθοδο την AdaNN NR One Pass όσον αφορά την διαχείριση του θορύβου. Όπως είναι ήδη γνωστό κατά την φάση του υπολογισμού του βέλτιστου κ για κάθε στιγμιότυπο υπάρχει ένα όριο μέχρι το οποίο θα προσπαθήσει να βρει ο αλγόριθμος το βέλτιστο κ. Όπως και στην προηγούμενη παραλλαγή αλλά και στην τρέχουσα παραλλαγή το όριο μέχρι το οποίο θα προσπαθήσει να υπολογίσει το βέλτιστο κ είναι η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων της κλάσης στην οποία ανήκει. Στην φάση του υπολογισμού των βέλτιστων κ, στις περιπτώσεις που δεν μπορεί να βρεθεί αυτό το βέλτιστο κ το στιγμιότυπο επισημαίνεται σαν ακατάλληλο και παίρνει την τιμή -1 που σημαίνει ότι είναι θόρυβος. Η διαφορά με την προηγούμενη παραλλαγή ξεκινάει από αυτό το σημείο και μετά. Στην προηγούμενη πρόταση θα γινόταν ένα πέρασμα στα στιγμιότυπα έτσι ώστε να αφαιρεθούν τα στιγμιότυπα τα οποία έχουν την τιμή -1. Σε αυτήν την παραλλαγή εφόσον γίνει το πρώτο πέρασμα θεωρούμε ότι εφόσον έχει αλλάξει το dataset, σημαίνει ότι έχουν τροποποιηθεί τα δεδομένα και μπορεί να έχουν αλλάξει οι τιμές των κ για διάφορα άλλα στιγμιότυπα και να υπάρχουν πλέον άλλα στιγμιότυπα τα οποία θα θεωρούνται θόρυβος. Συνεπώς θα πρέπει να γίνει επανυπολογισμός των κ όλων των στιγμιοτύπων και στην συνέχεια να γίνει ξανά αφαίρεση του θορύβου από τα στιγμιότυπα τα οποία. Αυτή η διαδικασία θα συνεχίσει να γίνεται για όσο θα υπάρχει θόρυβος στο σύνολο εκπαίδευσης. Με αυτήν την τακτική θεωρούμε ότι τα δεδομένα ελέγχου θα έχουν μεγαλύτερα ποσοστά σωστής κατηγοριοποίησης.



Σχήμα 3.5: Εφαρμογή AdaNN NR OnePass



Σχήμα 3.6: Εφαρμογή AdaNN NR MultiPass

Κεφάλαιο 4

Υλοποιήσεις σε Python

4.1 Η Γλώσσα Python

Στην παρούσα φάση της διπλωματικής, προκειμένου να γίνει πιο κατανοητή υπάρχει η ανάγκη για την ανάλυση της γλώσσας η οποία χρησιμοποιήθηκε για την υλοποίηση του προγράμματος της μηχανικής μάθησης, η οποία γλώσσα είναι η Python. Η Python είναι μια υψηλού επιπέδου γλώσσα η οποία είναι γενικού σκοπού και για αυτό τον λόγω έχει βρει εφαρμογή σε τομείς εντός και εκτός πληροφορικής. Γενικά η γλώσσα δεδομένου ότι υπάρχει αρκετά χρόνια στην αγορά, έχει χρησιμοποιηθεί για διάφορους σκοπούς. Συγκεκριμένα η γλώσσα χρησιμοποιείται για ανάπτυξη διαδικτυακών εφαρμογών, όσον αφορά την μεριά τους εξυπηρετητή. Αυτό συμβαίνει διότι μπορεί να υποστηρίξει σύνδεση με βάση δεδομένων, με χρήση των κατάλληλων βιβλιοθηκών καθώς και διαχείριση αρχείων. Επίσης μπορεί να χρησιμοποιηθεί για γενικότερη ανάπτυξη εφαρμογών σταθερού υπολογιστή, με γραφικό περιβάλλον και για προγραμματισμό συστημάτων. Ένα ιδιαίτερα σημαντικό σημείο είναι ότι η γλώσσα είναι κατάλληλη για επίλυση μαθηματικών προβλημάτων καθώς παρέχει μια πληθώρα βιβλιοθηκών η οποία θα μας δώσει διάφορες δυνατότητες οι οποίες κάνουν κατάλληλη την χρήση της Python για την ανάπτυξη της διπλωματικής. Η Python έχει υποστήριξη για διάφορες ενέργειες που είναι κατάλληλες για την μηχανική μάθηση, όπως διαχείριση πινάκων, πράξεις πινάκων, καθώς και μια πληθώρα από υλοποιήσεις αλγορίθμων τεχνητής νοημοσύνης οι οποίες μπορούν να αποτελέσουν την βάση για νέα προγράμματα. Επίσης σημαντικό σημείο της γλώσσας είναι ότι μπορεί να χειριστεί πολύ μεγάλο όγκο δεδομένων καθώς και να επιτελέσει σύνθετους μαθηματικούς υπολογισμούς σε αυτά.

Ένα τέτοιο χαρακτηριστικό παράδειγμα είναι η παρούσα διπλωματική η οποία θα χρησιμοποιήσει μια υπάρχουσα βιβλιοθήκη των κ εγγύτερων γειτόνων για την πολλαπλή εκτέλεση πολλαπλών πειραμάτων σε διάφορα σύνολα δεδομένων με σχετικά μεγάλο μέγεθος εκπαίδευσης δεδομένων αλλά και αξιολόγησης των αποτελεσμάτων των αλγορίθμων. Επίσης ιδιαίτερα όσον αφορά τους λόγους της επιλογής της γλώσσας είναι ότι μπορεί να εκτελεστεί σε διάφορες πλατφόρμες λογισμικού, όπως Windows, Macintosh καθώς και σε περιβάλλοντα Linux. Ένα άλλο σημαντικό χαρακτηριστικό της γλώσσας είναι ότι μπορεί να εκτελεστεί από διερμηνευτή (interpreter), πράγμα το οποίο σημαίνει ότι ο κώδικας ο οποίος γράφεται είναι άμεσα εκτελέσιμος. Αυτό το χαρακτηριστικό δίνει στον προγραμματιστή την ευελιξία για να εκτελέσει άμεσα κάποιες δικές του τροποποιημένες εντολές οποίες να χρειάζονται στην διερεύνηση του συνό-

λου δεδομένων, στην προ-επεξεργασία τους και στην εκπαίδευση του μοντέλου της μηχανικής μάθησης. Επίσης άξιο αναφοράς για την Python γλώσσα είναι ότι έχει σχετικά απλό σημαντικό συντακτικό το οποίο βασίζεται στο λεξιλόγιο της Αγγλικής γλώσσας πράγμα το οποίο είναι ιδιαίτερα σημαντικό και για νέους προγραμματιστές. Επίσης ιδιαίτερα σημαντικό χαρακτηριστικό είναι ότι λόγω των πολλαπλών βιβλιοθηκών που παρέχει καθώς και του απλού συντακτικού της επιτρέπει σε προγραμματιστές να γράψουν προγράμματά με πολύ λιγότερες γραμμές κώδικα. Αυτό επιτυγχάνεται χάρη σε πολλά παραδείγματα χρήσης της στο Διαδίκτυο με διάφορες μεθόδους ανάπτυξης λογισμικού όπως για παράδειγμα τον διαδικαστικό προγραμματισμό και τον δομημένο προγραμματισμό, αλλά και τον αντικειμενοστραφή. Μια ιδιαίτερη διαφορά της γλώσσας σε σχέση με άλλες γλώσσες προγραμματισμού είναι ότι είναι δυναμικού τύπου, δηλαδή οι μεταβλητές λαμβάνουν τον κατάλληλο τύπο από μόνες τους ανάλογα με το περιεχόμενο το οποίο αποθηκεύουν. Γενικά η γλώσσα της Python δίνει βάση στην αναγνωσιμότητα της καθώς σε αντίθεση με άλλες γλώσσες προγραμματισμού οι εντολές της δεν καταλήγουν σε ελληνικό ερωτηματικό ";", δεν έχει άγκιστρα για τον διαχωρισμό των διάφορων block κώδικα και έχει ως βάση την στοίχιση του κώδικα. Αυτό φαίνεται στο γεγονός ότι η ορατότητα των μεταβλητών του κώδικα και το που μπορούν να χρησιμοποιηθούν βασίζονται στα κενά της στοίχιση κάθε γραμμής. Για παράδειγμα αν μια εντολή βρίσκεται κατά ένα ειδικό χαρακτήρα tab πιο δεξιά από την πάνω γραμμή, τότε ο διερμηνευτής αντιλαμβάνεται ότι μεταβλητή έχει άλλη ορατότητα (π.χ. μια τοπική μεταβλητή μια θηλιάς while). Αυτή η δήλωση σε μια άλλη γλώσσα όπως η Java που έχει ως βάση την προγραμματιστική γλώσσα C, θα γινόταν με την χρήση των κατάλληλων άγκιστρων ορατότητας. Χάρη σε αυτόν τον κανόνα της στοίχισης προσπαθεί να δώσει λύση στην αναγνωσιμότητα του κώδικα.

Παρά το γεγονός και της ολοένα αυξανόμενης τάσης της χρήσης της γλώσσας κυρίως σε τομείς της μηχανικής μάθησης αλλά και για εκπαιδευτικούς σκοπούς για πιο νέους προγραμματιστές ή νέους στον χώρο της πληροφορικής, πράγμα το οποίο μπορεί να δώσει μια αίσθηση ότι η γλώσσα είναι καινούργια σε σχέση με άλλες ευρέως χρησιμοποιούμενες γλώσσας στην αγορά, η ιστορία της Python ξεκινάει αρκετά παλιά και συγκεκριμένα στο 1991 [19]. Ιστορικά, η δημιουργία της Python έγινε το 1990 από τον Guido van Rossum στην Ολλανδία η οποία διαδέχτηκε την γλώσσα ABC. Ενδεικτικό είναι ότι η γλώσσα παρόλο που έχει πάρα πολλά άτομα τα οποία συνεργάστηκαν για την υλοποίηση της, βασικός της δημιουργός και συγγραφές θεωρείται ο ο Guido ο οποίος ήταν ο συνεχιστής της δημιουργίας της Python μαζί με διάφορους εθελοντές που συνεργάστηκαν. Η συγγραφή της γλώσσας συνεχίστηκε το 1995 καθώς και το 2000 στην Virginia των ΗΠΑ όπου ανακοινώθηκαν πολλές εκδόσεις της. Η πρώτη έκδοση της γλώσσας η οποία δημιουργήθηκε και ανακοινώθηκε είναι η έκδοση Python 0.9.0 με ημερομηνία έναρξης το 1991. Η έκδοση Python 2 ανακοινώθηκε το 2000 και γρήγορα έγινε διάσημη και η έκδοση Python 3 ανακοινώθηκε το 2008. Το 2000 ο Guido καθώς και η κεντρική ομάδα της δημιουργίας και ανάπτυξης της Python θα μεταφερθούν στην ιστοσελίδα BeOpen.com για τον σχηματισμό της ομάδας PythonLabs και στην συνέχεια στην μεταφορά στην ομάδα Digital Creations. Το 2001 η Python Software Foundation σχηματίστηκε απλά για την ιδιοκτησία της Python ως ένας μη κερδοσκοπικός οργανισμός για την ανάπτυξη όλης της τεχνολογίας του ανοιχτού λογισμικού που σχετίζεται με την προγραμματιστική γλώσσα Python . Άξιο αναφοράς είναι πως όλες οι εκδόσεις της προγραμματιστικής γλώσσας Python ανήκουν στο ανοιχτό λογισμικό. Οι εκδόσεις αυτές υλοποιήθηκαν κάτω από τις οδηγίες και την καθοδήγησή του Guido ο οποίος είχε στην διάθεση τους διάφορους εξωτερικούς εθελοντές για την πραγματοποίηση του

πρότζεκτ. Από εκείνο το σημείο και μετά η γλώσσα της Python ανακοίνωσε διάφορες εκδόσεις στην αγορά οι οποίες είχαν ως σκοπό την δημιουργία βελτιώσεων αλλά και επίλυση διάφορων προβλημάτων ασφάλειας τα οποία προέκυπταν ανά τις εκδόσεις. Στην φάση της συγγραφής της διπλωματικής, η τελευταία έκδοση η οποία κυκλοφορεί και είναι σταθερή ανακοινώθηκε στις 24 Νοέμβριου του 2022 και είναι η 3.11 με σημαντικές βελτιώσεις της ταχύτητας εκτέλεσης.

4.2 Η Βιβλιοθήκη scikit learn

Σε αυτό το σημείο είναι πολύ σημαντικό να αναφερθούμε στην βιβλιοθήκη η οποία χρησιμοποιήθηκε ως βάση για την δημιουργία των παραλλαγών του Adaptive-NN την scikit-learn. Η scikit-learn είναι μια διάσημη βιβλιοθήκη για μηχανική μάθηση αλλά και εξόρυξη δεδομένων σε Python. Σημαντικό χαρακτηριστικό είναι ότι και αυτή η βιβλιοθήκη είναι ανοιχτού λογισμικού οπότε χρησιμοποιείται ευρέως για εκπαιδευτικούς σκοπούς εφόσον είναι προσβάσιμη από όλους και μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς. Η βιβλιοθήκη παρέχει ένα σύνολο από αλγορίθμους αλλά και διάφορα εργαλεία για προ-επεξεργασία δεδομένων, εφαρμογή και εκπαίδευση μοντέλων μηχανικής μάθησης, επιλογή χαρακτηριστικών καθώς και εκτέλεση αξιολόγησης των αποτελεσμάτων των διάφορων αλγορίθμων αυτών. Μπορεί να υποστηρίξει συγκεκριμένα προβλήματα classification και regression όπως επίσης και προβλήματα συσταδοποίησης. Συγκεκριμένα στην παρούσα διπλωματική θα γίνει χρήση του αλγορίθμου K-NN δηλαδή του ταξινομητή K εγγύτερων γειτόνων ο οποίος θα προσαρμοστεί για να λειτουργήσει με ένα μεταβλητό K. Το λογισμικό του scikit-learn είναι βασισμένο σε άλλες προϋπάρχουσες βιβλιοθήκες όπως h NumPy η SciPy αλλά και η matplotlib. Η βιβλιοθήκη του Scikit Learn χρησιμοποιείται γενικά και στην ακαδημαϊκή κοινότητα αλλά και στην βιομηχανία, λύνοντας προβλήματα ταξινόμησης εικόνων, επεξεργασία φυσικής γλώσσας αλλά και πρόβλεψη αποτελεσμάτων με βάση κάποιο μοντέλο μηχανικής μάθησης καθώς είναι μια βιβλιοθήκη εύκολη στην χρήση αλλά με ισχυρή παραμετροποίηση και λειτουργικότητα σε ένα μεγάλο σύνολο αλγορίθμων.

4.3 Η Βιβλιοθήκη Numpy

Σε αυτό το σημείο είναι ιδιαίτερα σημαντικό να γίνει αναφορά σε μια άλλη βασική βιβλιοθήκη η οποία θα είναι ιδιαίτερά χρήσιμη στα πειράματα τα οποία θα εκτελεστούν. Η βιβλιοθήκη η οποία είναι πολύ χρήσιμη είναι η numpy η οποία χρησιμοποιείται ευρέως σε προγράμματα python και ιδιαίτερα προβλήματα μηχανικής μάθησης [20]. Η βιβλιοθήκη αυτή είναι ένα πρότζεκτ ανοιχτού λογισμικού το οποίο επιτρέπει μια πληθώρα αριθμητικών υπολογισμών με τη προγραμματιστική γλώσσα Python. Η γλώσσα αυτή δημιουργήθηκε το 2005 και αποτέλεσε την πρώιμη βάση για τις βιβλιοθήκες Numeric και Numarrary. Το πρότζεκτ της Numpy είναι εξ ολοκλήρου ένα πρότζεκτ ανοιχτού λογισμικού και είναι ελεύθερο προς χρήση για όλους.Η NumPy ή αλλιώς γνωστή σαν Numeric Python είναι ένα επιστημονικό πακέτο για το οποίο χρησιμοποιείται για επιστημονικούς υπολογισμούς, όπως οι πράξεις πινάκων πράγμα το οποίο είναι ιδιαίτερα χρήσιμο για επίλυση μαθηματικών προβλημάτων.
Η βιβλιοθήκη numpy της Python είναι ιδιαίτερα παλιά βιβλιοθήκη καθώς η αρχή της έγινε το

1995 από τον Jim Hugunin με την βοήθεια διάφορων άλλων προγραμμάτων. Σε αυτή την φάση η βιβλιοθήκη ακόμα ονομαζόταν Numeric και ήταν ο πρόγονος της σημερινής NumPy. Στην συνέχεια το 2005 φτιάχτηκε η NumPy η οποία χάρη στον Tavis Oliphan χρησιμοποίησε στοιχεία από τον προηγούμενη βιβλιοθήκη την Numeric αλλά και συνδυάζοντάς χαρακτηριστικά της ανταγωνιστικής της βιβλιοθήκης Numeric.Χάρη στον Travis Oliphan, έναν προγραμματιστή της βιβλιοθήκης Numpy επετεύχθη η δημιουργία μιας κοινότητας η οποία θα ήταν πίσω από την βιβλιοθήκη Numpy η οποία θα έκρυβε μια συνολική μαθηματική λειτουργικότητα των πινάκων. Όλη αυτή η λειτουργικότητα θα παρουσιαζόταν με την πρώτη παρουσίαση και έκδοση τη NumPy το 2006.

Σε αυτό το σημείο εφόσον έχει γίνει μια γενικότερη εισαγωγή στην Numpy καθώς και μια μικρή ιστορική αναδρομή είναι σημαντικό να γίνει αναφορά στους λόγους για τους οποίους χρησιμοποιήθηκε η βιβλιοθήκη Numpy της Python. Αυτοί οι λόγοι μπορεί να ποικίλουν όσον αφορά την φύση του προβλήματος αλλά και την ταχύτητά επίλυσης του προβλήματος.
Το πρώτο σημαντικό σημείο έχει να κάνει με την φύση του προβλήματος, το οποίο δεν είναι άλλο από την επίλυση προβλήματος κατηγοριοποίησης το οποίο ανήκει στο ευρύτερο πλαίσιο των επιστημονικών εφαρμογών .Αυτή η δυνατότητα επίλυσης επιστημονικών προβλημάτων από την βιβλιοθήκη NumPy είναι που θα αποτελέσει την βάση για την υλοποίηση του αλγορίθμου AdaNN μέσα στον οποίο θα πρέπει να γίνουν κάποιες πράξεις πινάκων προκειμένου να εξαχθεί το βέλτιστο αποτέλεσμα. Αυτή λοιπόν η βιβλιοθήκη Python περιλαμβάνει μια πληθώρα λειτουργιών όπως η παροχή πολυδιάστατων πινάκων έως διακριτούς μετασχηματισμούς Φουριέ, γραμμική άλγεβρα και βασικές λειτουργίες στατιστικής. Ιδιαίτερα πολύ σημαντικό σημείο στο οποίο θα πρέπει να γίνει αναφορά είναι ότι χάρη στην βιβλιοθήκη NumPy μπορεί να γίνουν κάποιοι βασικοί μετασχηματισμοί στις βασικές δομές αποθήκευσης των δεδομένων του προβλήματος που θα επιλύσει ο AdaNN. Οι βασικές μορφές αποθήκευσης δεδομένων που θα χρησιμοποιηθούν στο πρόγραμμα του AdaNN είναι κυρίως πίνακες, λίστες αλλά και δομές αποθήκευσης tuple. Ένα άλλο χαρακτηριστικό της βιβλιοθήκης NumPy είναι οι δομές πολλών διαστάσεων οι οποίες είναι χρήσιμες καθώς στην περίπτωση της διπλωματικής θα πρέπει να γίνει εκπαίδευση με τον AdaNN και θα πρέπει να γίνει εφαρμογή του κατηγοριοποιητή σε δεδομένα τα οποία έχουν ένα μεταβλητό αριθμό στηλών οι οποίες είναι και το πλήθος των χαρακτηριστικών που θα έχει το κάθε σύνολο δεδομένων. Για αυτόν τον λόγο, θεωρώντας ότι κάθε μια στήλη είναι ένα χαρακτηριστικό αναπαριστούμε κάθε μια από αυτές σε μια διαφορετική διάσταση σε έναν πίνακα. Για παράδειγμα εάν σε ένα σύνολο δεδομένων υπάρχουν 3 χαρακτηριστικά, θα χρειαστεί να απεικονιστούν σε ένα σύστημα συντενταγμένων το οποίο θα έχει τους 3 άξονες x,y,z (τετμημένη, τεταγμένη, κατηγμένη). Η βιβλιοθήκη NumPy παρέχει μια δομή αποθήκευσης που λέγεται ndarray το οποίο συνήθως θα έχει ένα προκαθορισμένο μέγεθος από στοιχεία τα οποία έχουν τον ίδιο τύπο και το ίδιο μέγεθος. Ο αριθμός των διαστάσεων και των στοιχείων αναπαρίστανται από ένα tuple μη αρνητικούς αριθμούς οι οποίοι θα καθορίσουν για κάθε διάσταση το μέγεθος της. Τέλος είναι σημαντικό να αναφερθεί ότι η βιβλιοθήκη NumPy είναι ιδιαίτερα χρήσιμη διότι θα χρειαστεί να γίνουν μαζικές προσπελάσεις πινάκων, πράξεις πινάκων και λιστών καθώς και μια άλλη σειρά διεργασιών όπως ταξινόμηση, επιλογή αλλά και λογική τροποποίηση των δεδομένων του προβλήματος στις οποίες η βιβλιοθήκη μπορεί να αντεπεξέλθει με καλές ταχύτητες.
Σε αυτό το σημείο είναι σημαντικό να γίνει κατανοητό ότι καθώς μιλάμε για μια εκπαίδευση ενός κατηγοριοποιητή και γενίκευση του μοντέλου θα πρέπει να λάβουμε υπόψη γενικά και

τους χρόνους για τους οποίους θα μπορεί να εκπαιδευτεί αλλά και να γενικεύσει [21]. Σε αυτό το σημείο πρέπει να αναφερθεί ότι η βιβλιοθήκη NumPy έχει υψηλή απόδοση καθώς επιλύει το πρόβλημα της καθυστέρησης παρέχοντας πολυδιάστατους πίνακες συναρτήσεις αλλά και διάφορους τελεστές οι οποίοι λειτουργούν πάνω σε πίνακες ιδιαίτερα αποδοτικά. Άξιο αναφοράς είναι ότι οι πίνακες NumPy είναι μια συλλογή από δεδομένα τα οποία έχουν τον ίδιο τύπο και είναι πιο πυκνά τοποθετημένα στην μνήμη. Επίσης σημαντικό είναι ότι η NumPy έχει την δυνατότητα της παράλληλης επεξεργασίας, δηλαδή να πάρει μια πιο σύνθετη διαδικασία και να κάνει μια διαίρεση αυτής σε πολλές υπό διαδικασίες οι οποίες θα μπορέσουν να εκτελεστούν παράλληλα. Αυτό μπορεί να έχει ως αποτέλεσμα μια διαδικασία η οποία θα εκτελεστεί από την βιβλιοθήκη NumPy να μπορεί να εκτελεστεί από 5 έως 100 φορές γρηγορότερα από μια κανονική λίστα της Python . Επίσης άλλο σημείο στο οποίο πρέπει να σταθούμε είναι πως η βιβλιοθήκη numpy παρέχει εργαλεία για την ενσωμάτωση κώδικα από άλλες γλώσσες προγραμματισμού όπως η C, η C++ αλλά και η Fortran πράγμα το οποίο και εξηγεί τον λόγο επιλογής της γλώσσας του πηγαίου κώδικα στις βιβλιοθήκες που παρέχει η NumPy. Άξιο αναφοράς είναι ότι οι συναρτήσεις οι οποίες είναι υλοποιημένες και παρέχονται στον developer έχουν τον πηγαίο κώδικα τους γραμμένο στην γλώσσα C η οποία έχει σαν εγγενές χαρακτηριστικό την ταχύτητα της. Αυτό το στοιχείο κάνει την βιβλιοθήκη να υπερτερεί από τις άλλες συναρτήσεις της Python όπως αυτές των λιστών. Τέλος σημαντικό να αναφερθεί είναι πως οι λίστες τις Python έχουν πιο μεγάλη καθυστέρηση στην εκτέλεση καθώς μπορεί να αποθηκεύουν δεδομένα τα οποία είναι διαφορετικού τύπου δεδομένων πράγμα το οποίο προσδίδει κάποιους έξτρα περιορισμούς κατά την φάση των μαζικών πράξεων πάνω στα στοιχεία των λιστών.

4.4 AdaNN και παραλλαγές

4.4.1 Παραλλαγές Ορίου Βέλτιστου κ Χωρίς Αφαίρεση Θορύβου

Σε αυτό το σημείο θα πρέπει να γίνει ιδιαίτερη αναφορά στον κώδικα ο οποίος έχει γραφτεί τόσο για την απλή υλοποίηση του αλγορίθμου AdaNN καθώς και των παραλλαγών του. Σε αυτήν την πρώτη παραλλαγή που θα αναλυθεί παρακάτω υπάρχει και η υλοποίηση του απλού AdaNN. Στον κώδικα που θα παρατεθεί γίνεται ανάλυση των διάφορων παραλλαγών του AdaNN με διαφορετικά όρια στα οποία θα φτάσει για να υπολογίσει το βέλτιστο κ ο AdaNN καθώς και το σταθερό όριο το οποίο ισούται με την τιμή 9. Όπως φαίνεται παρακάτω υπάρχει μια κλάση AdaNN η οποία θα αναλάβει την παραγωγή των κ για κάθε στιγμιότυπο που υπάρχει στο σύνολο δεδομένων. Το σύνολο των δεδομένων είναι ένα αρχείο csv το οποίο έχει σε κάθε γραμμή τα χαρακτηριστικά και την κλάση που ανήκει το χαρακτηριστικό. Μετά από την επεξεργασία αυτού του αρχείου θα πρέπει να δημιουργηθεί το ίδιο αρχείο απλώς μετά από την κατηγορία του στιγμιότυπου θα πρέπει να υπάρχει και η τιμή του βέλτιστου κ για εκείνο το στιγμιότυπο.
Στο παρακάτω κομμάτι κώδικα 4.1 φαίνεται η συνάρτηση init με την οποία θα οριστεί το είδος της συνάρτησης υπολογισμού ορίου που θα χρησιμοποιηθεί. Επίσης βλέπουμε πως ορίζουμε σαν προκαθορισμένη τιμή την const, δηλαδή η συνάρτηση ορίου θα υπολογίζει το μέγιστο όριο του βέλτιστου κ με την σταθερή συνάρτηση η οποία έχει τιμή 9 και είναι η απλή εκδοχή του αλγορίθμου AdaNN. Στην συνάρτηση υπολογισμού του μέγιστου βέλτιστου κ θα περάσουμε τον αριθμό των στιγμιοτύπων έτσι ώστε ανάλογα με το είδος της συνάρτησης που έχουμε περάσει

στην κλάση να χρησιμοποιήσει διαφορετικό τρόπο υπολογισμού του ορίου. Στην δομή ελέγχου βλέπουμε πως υπάρχει η συνάρτηση sqrt που δείχνει την τετραγωνική ρίζα του αριθμού των στιγμιοτύπων, μετά είναι η cbrt όπου είναι η κυβική ρίζα του αριθμού των στιγμιοτύπων και η τέλος η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων της κλάσης του στιγμιότυπου που εξετάζεται. Τέλος βλέπουμε πως αν δεν υπάρχει κάτι από τα παραπάνω ορισμένα θα γίνει χρήση της σταθερή συνάρτησης επιστρέφοντας την τιμή 9.

```
class AdaNN:


    def __init__(self, k_limit_fuction = 'const'):
        self.k_limit_fuction = k_limit_fuction
        print('\t k limit function set = ' + k_limit_fuction)



    def calculateMaximumKLimit(self, y_test, y_train, numberOfInstances):

        if self.k_limit_fuction == 'sqrt' :
            return  round(math.sqrt(numberOfInstances))
        elif self.k_limit_fuction == 'cbrt' :
            return round(numberOfInstances ** (1/3))
        elif self.k_limit_fuction == 'sqrt_limit_class' :

            testingInstanceClass =  y_test[0];
            return round(np.sqrt(y_train.count(testingInstanceClass)) + 1)
        else:
            return  9
```

Σχήμα 4.1: Συνάρτηση Υπολογισμού Μέγιστης Τιμής Βέλτιστου Κ

Το αμέσως επόμενο σημείο το οποίο θα αναλυθεί είναι η μέθοδος η οποία εν τέλει θα υπολογίστει το βέλτιστο κ για κάθε στιγμιότυπο του συνόλου δεδομένων. Στο παρακάτω παράδειγμα 4.2 βλέπουμε πως η μέθοδος calculateBestK θα λάβει 4 πίνακες. Οι πρώτοι 2 θα είναι οι xtest, ytest, όπου είναι τα χαρακτηριστικά του στιγμιότυπου που υπολογίζουμε το βέλτιστο και η κλάση η οποία ανήκει. Από την άλλη xtrain, ytrain είναι όλα τα άλλα στιγμιότυπα του συνόλου εκπαίδευσης και οι στόχοι τους. Στην συνέχεια θα γίνει υπολογισμός του ανώτατου ορίου που θα μπορεί να έχει το βέλτιστο κ. Αμέσως μετά ακολουθεί η διαδικασία εύρεσης του βέλτιστου κ. Για αρχή θα δοθεί η τιμή 1 στην μεταβλητή k η οποία θα έχει τον αριθμό των γειτόνων που θα προσπαθήσει εκτελέσει διαδοχικά αλγορίθμους KNN κ αυξάνοντας τον αριθμό των γειτόνων κατά 1 κάθε φορά. Για όσο μπορεί θα εκτελεί αλγορίθμους K εγγύτερων γειτόνων όπου θα εκπαιδεύεται με τα υπόλοιπα δεδομένα της εκπαίδευσης και θα προσπαθεί να προβλέψει για το τρέχον στιγμιότυπο στο οποίο γίνεται ο υπολογισμός του βέλτιστου κ. Σε περίπτωση που η ακρίβεια ισούται με 1 δηλαδή πετύχει να μαντέψει σωστά την κατηγορία του εξεταζόμενου στιγμιότυπου θα σταματήσει την εκτέλεση των διαδοχικών αλγορίθμων KNN και θα επιστρέψει το βέλτιστο κ. Σε άλλη περίπτωση που αλγόριθμος φτάσει στο όριο του μέγιστου ορίου του βέλτιστου κ θα γίνει παύση της αναζήτησης θα επιστρέφει η τελευταία τιμή. Συνεπώς είτε ο αλγόριθμος θα βρει αυτό το βέλτιστο κ το οποίο ψάχνουν είτε θα φτάσει στο ανώτατο του όριο και θα επιστραφεί σαν βέλτιστο κ το ανώτατο όριο το οποίο μπορεί να φτάσει το βέλτιστο κ.

Η αμέσως επόμενη μέθοδος που θα αναφερθεί είναι αυτή που τελικά θα χρησιμοποιήσει τις προηγούμενες μεθόδους οι οποίες υλοποιήθηκαν 4.3. Αρχικά αυτή η μέθοδος θα δεχτεί την

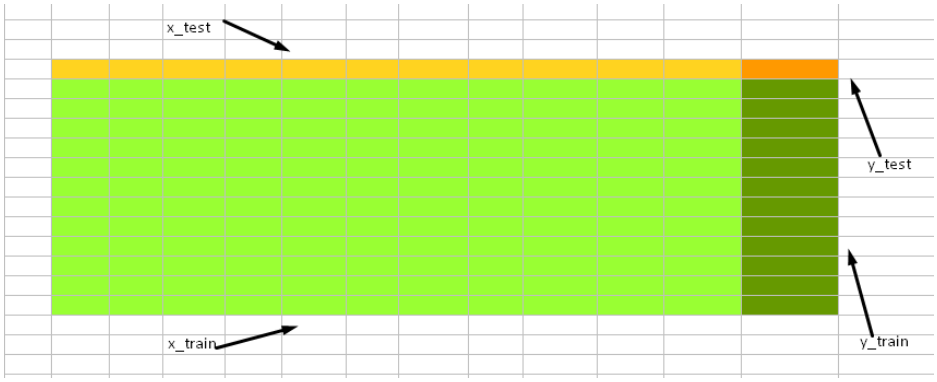
```
def calculateBestK(self, x_train, y_train, x_test, y_test):
    y_test = [y_test]
    x_test = x_test.reshape(1,-1)

    numberOfTrainingInstances = len(x_train)

    k_limit = self.calculateMaximumKLimit(y_test, y_train, numberOfTrainingInstances)

    print('\t \t k limit = ' + str(k_limit))

    k = 1
    while True:

        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(x_train, y_train)
        y_pred = knn.predict(x_test)

        if metrics.accuracy_score(y_test, y_pred) == 1.0 :
             break

        if k == self.calculateMaximumKLimit(y_test, y_train, numberOfTrainingInstances):
            return self.calculateMaximumKLimit(y_test, y_train, numberOfTrainingInstances)
            break
        k+=1
    return k
```

Σχήμα 4.2: Μέθοδος υπολογισμού του βέλτιστου κ

διαδρομή από τον φάκελο στον οποίο βρίσκονται τα αρχεία csv, τα οποία είναι τα σύνολα εκπαίδευσης καθώς και το όνομα του αρχείου για το οποίο θα κάνει τον υπολογισμό του βέλτιστου κ. Επίσης άξιο αναφοράς είναι ότι ζητείται και σαν παράμετρος στην μέθοδο ο διαχωριστής με βάση τον οποίο χωρίζονται τα χαρακτηριστικά και οι κλάσεις στα σύνολα εκπαίδευσης. Έτσι θα γίνει ανάγνωση των δεδομένων των συνόλων εκπαίδευσης τα οποία θα χωριστούν στους εξής πίνακες x,y όπου ο πρώτος έχει τα χαρακτηριστικά των δεδομένων εκπαίδευσης και ο δεύτερος έχει τις κλάσεις στις οποίες ανήκουν αυτά τα στιγμιότυπα. Αμέσως μετά θα γίνει δημιουργία ενός καινούργιου αρχείου csv το οποίο θα έχει ακριβώς την ίδια δομή με τα αρχεία εκπαίδευσης με την διαφορά ότι θα έχει μια ακόμα στήλη για το βέλτιστο κ ανά στιγμιότυπο. Συνεπώς αυτό το οποίο θα γίνει είναι να γίνει διάβασμα του αρχείου της εκπαίδευσης και με μια δομή επανάληψης να γίνει χωρισμός του dataset στα εξής 4 μέρη τα οποία μας χρειάζονται. Αμέσως μετά θα γίνει ο υπολογισμός του βέλτιστου κ με την μέθοδο που αναλύσαμε παραπάνω και πλέον εφόσον έχουμε και τα χαρακτηριστικά του στιγμιότυπου και την κατηγορία στην οποία ανήκει αλλά και το βέλτιστο κ το μόνο που μένει είναι να γραφτούν στα καινούργιο αρχείο το οποίο θα έχει εκτός από την έξτρα στήλη θα έχει και το ίδιο όνομα με το αρχείο εκπαίδευσης αλλά θα έχει το πρόθεμα 'k_'. Τέλος το μόνο το οποίο απομένει είναι να γραφτούν τα δεδομένα στο αρχείο καθώς και να κλείνει το stream για το αρχείο στο οποίο γράφουμε.Για να γίνει πιο κατανοητή η δομή των δεδομένων στους λογικούς πίνακες που αναφέρθηκαν παρουσιάζεται μια λογική απεικόνιση του εξεταζόμενου στιγμιότυπου σε σχέση με τα υπόλοιπα δεδομένα στο σχήμα 4.4.

```
def method(self, folder, trainingDataSetFileName, dSep):

    data = read_csv(folder + trainingDataSetFileName, header=None, sep = dSep).values


    x=data[:, 0:data.shape[1]-1].astype(float) #get all but last column, (attributes)
    y=data[:, data.shape[1]-1] #get last column,  targets

    datasetWithK = open(folder + str("k_") + trainingDataSetFileName, "a")


    i = 0

    while i < x.shape[0]: #number of instances
        x_train = list(x)[:i] + list(x)[i+1:]
        x_test = x[i]
        y_train = list(y)[:i] + list(y)[i+1:]
        y_test = y[i]

        k = self.calculateBestK(x_train, y_train, x_test, y_test);

        newLine = ""
        for j in range(x.shape[1]):
            newLine += str(x_test[j]) + dSep
        newLine += str(int(y[i])) + dSep + str(k) + "\n"
        datasetWithK.write(newLine)

        print(i)

        i=i+1



    datasetWithK.close()

    print('\t Done' + folder + '\n')
    get_ipython().magic('reset -sf')
```

Σχήμα 4.3: Μέθοδος Διαχείρισης Αρχείου και υπολογισμού Βέλτιστου K



Σχήμα 4.4: Χωρισμός δεδομένων

Σε αυτό το σημείο εφόσον έχει γίνει η επεξεργασία για κάθε ένα στιγμιότυπο και έχει υπολογιστεί το βέλτιστο κ και έχει δημιουργηθεί το αρχείο csv το οποίο θα έχει την επιπλέον στήλη που περιλαμβάνει το κάθε βέλτιστο κ θα πρέπει να γίνει εξαγωγή συμπεράσματος για κάθε στιγμιότυπου του συνόλου δεδομένων. Παρακάτω θα αναλυθεί η μέθοδος 4.5 η οποία θα εξάγει συμπέρασμα για κάθε στιγμιότυπου του συνόλου ελέγχου. Αρχικά θα πρέπει να γίνει αναφορά στις παραμέτρους που δέχεται αυτή η μέθοδος. Η πρώτη παράμετρος που παίρνει παραμετρικά είναι ο φάκελος στον οποίο βρίσκεται το αρχείο με το modelFileName το οποίο είναι το csv με τα βέλτιστα κ για κάθε στιγμιότυπο. Στην συνέχεια θα γίνει διάβασμα των διαφορετικών στηλών από το αρχείο με τα βέλτιστα κ. Το αποτέλεσμα του διαβάσματος αυτού είναι ένας πίνακας xModel, ένας πίνακας yModel και ένας πίνακας k. Και οι τρεις πίνακες έχουν

τον ίδιο αριθμό γραμμών που ισούνται με τον αριθμό των γραμμών του αρχείου. Ο πρώτος πίνακας θα περιλαμβάνει τα χαρακτηριστικά όλων των στιγμιοτύπων. Ο δεύτερος πίνακας θα έχει τους στόχους που έχουν χρησιμοποιηθεί στην φάση της εκπαίδευσης. Κάτι ανάλογο θα γίνει και στο αρχείο με δεδομένα του ελέγχου που θα χρησιμοποιηθούν για τον έλεγχο της απόδοσης του αλγορίθμου. Αυτό που γίνεται είναι ότι όλες οι στήλες εκτός από την τελευταία είναι τα χαρακτηριστικά των στιγμιοτύπων που θα χρησιμοποιηθούν για τον έλεγχο δηλαδή ο πίνακας xTestData και τέλος θα χρησιμοποιηθούν και οι στόχοι του συνόλου ελέγχου για την αξιολόγηση οι οποίοι είναι αποθηκευμένοι στον πίνακα yTestData. Αυτό το οποίο θα γίνει μετά είναι μια δομή επανάληψης η οποία θα κάνει προσπέλαση όλα τα στιγμιότυπα από το σύνολο ελέγχου και για κάθε ένα από αυτά θα βρίσκει το κοντινότερο του στιγμιότυπο από το σύνολο εκπαίδευσης. Αυτό που επιστρέφεται πραγματικά είναι ο δείκτης του στιγμιότυπου αυτού που βρίσκεται στο σύνολο δεδομένων εκπαίδευσης και για αυτό τον τον δείκτη του θα πάρουμε το k του στιγμιότυπου της εκπαίδευσης. Άρα σε αυτήν την φάση με δεδομένο ένα στιγμιότυπο ελέγχου έχει βρεθεί το κοντινότερο του από το σύνολο εκπαίδευσης και έχει βρεθεί την τιμή του κ που θα δανειστεί. Στην συνέχεια θα γίνει εκτέλεση του αλγορίθμου κ εγγύτερων γειτόνων με την επιλεχθείσα τιμή του κ. Στην συνέχεια θα γίνει εκτέλεση της εκπαίδευσης και της εξαγωγής του αποτελέσματος για το τρέχον στιγμιότυπο ελέγχου και το αποτέλεσμα θα είναι στην μεταβλητή y_pred. Χάρη στην μέθοδο precision_recall_fscore_support για τις προβλέψεις που έχουν γίνει με τους πραγματικούς στόχους των στιγμιοτύπων του συνόλου ελέγχου. Τέλος σε μια μορφή δεδομένων tuple θα γίνει υπολογισμός των μέσων τιμών των μετρικών precision, recall, fscore, support και του accuracy. Αυτά θα επιστραφούν στην main μέθοδο για να γραφτούν στα συνολικά αποτελέσματα του αλγορίθμου.

```
def method1(folder, testDatasetName, modelFileName, dSep):
    #every line in modelFile has this format x1 x2  x3  x4  y   k
    #modelFileName = "k_data1"
    #every line in testFile has this format x1  x2  x3  x4  y
    #testDatasetName = "data_te1"

    #dSep = '   '



    modelData = read_csv(folder + modelFileName, header=None, sep = dSep).values
    testData = read_csv(folder + testDatasetName, header=None, sep = dSep).values

    xModel = modelData[:, 0:modelData.shape[1] - 2].astype(float)
    yModel = modelData[:, modelData.shape[1] - 2].astype(int)
    k = modelData[:, modelData.shape[1] - 1].astype(int)

    xTestData = testData[:, 0:testData.shape[1] -1].astype(float)
    yTestData = testData[:, testData.shape[1] -1].astype(int)

    predictionsForTestData = list()

    for xTestInstance in xTestData:
        xTestInstance = xTestInstance.reshape(1,-1) #transform array from 2D to 1D

        nearestNeigborIndex = find_indexOf_nearest_instance_of_model(xTestInstance, xModel)

        #We should not allow k greater than the number of instances of model
        kForTestInstance = k[nearestNeigborIndex] if k[nearestNeigborIndex] < k.shape[0] else k.shape[0]

        knn = KNeighborsClassifier(n_neighbors=kForTestInstance, weights = my_weight_func)
        knn.fit(xModel, yModel)

        y_pred = knn.predict(xTestInstance)
        predictionsForTestData.append(y_pred)

    precision, recall, fscore, supp = metrics.precision_recall_fscore_support(yTestData, predictionsForTestData, average=None)

    scores_tuple = (np.mean(precision),
                    np.mean(recall),
                    np.mean(fscore),
                    np.mean(supp),
                    metrics.accuracy_score(yTestData, predictionsForTestData))

    return scores_tuple
```

Σχήμα 4.5: Μέθοδος Αξιολόγησης μοντέλου

Επίσης άξιο αναφοράς είναι ο τρόπος που υπάρχει για να βρεθεί το κοντινότερο στιγμιότυπο του συνόλου εκπαίδευσης σε ένα δοθέν στιγμιότυπο ελέγχου. Για αρχή θα περαστούν σαν παράμετρος τα χαρακτηριστικά του στιγμιότυπου ελέγχου και ο πίνακας και με τα χαρακτηρι-

στικά του συνόλου εκπαίδευσης από το οποίο θα γίνει η ανάκτηση του ζητούμενου δείκτη. Θα γίνει εκτέλεση εγγύτερου γείτονα και στην συνέχεια θα γίνει εκπαίδευση με τον πίνακα των χαρακτηριστικών του συνόλου εκπαίδευσης και μετά θα γίνει επιστροφή του δείκτη του κοντινότερου στιγμιότυπου.

```
def find_indexOf_nearest_instance_of_model(xTestInstance, xModel):
    neigh = NearestNeighbors(n_neighbors=1)
    neigh.fit(xModel)
    index = neigh.kneighbors(xTestInstance, return_distance=False)
    return index[0][0]
```

Σχήμα 4.6: Μέθοδος εύρεσης δείκτη κοντινότερου γείτονα συνόλου εκπαίδευσης

Σημαντικό σημείο στο οποίο θα πρέπει να σταθούμε είναι η διαχείριση ισοψηφιών. Το πρόβλημα προκύπτει όταν γίνονται διαδοχικές εκτελέσεις αλγορίθμων KNN αυξάνοντας το κ μπορεί να προκύψουν ισοψηφίες μεταξύ εγγύτερων γειτόνων. Στην πραγματικότητα αυτό όμως δεν είναι απαραίτητα σωστό, καθώς ένα στιγμιότυπο το οποίο είναι πιο κοντά θα πρέπει να έχει μεγαλύτερη βαρύτητα στην λήψη απόφασης. Για αυτόν τον λόγο και θα πρέπει να γίνει ανάθεση μεγαλύτερου βάρους από τους άλλους γείτονες. Για αυτόν τον λόγο γίνεται αρχικοποίηση ενός πίνακας από μονάδες και στον κοντινότερο γείτονα θα γίνει ανάθεση της τιμής 1.1

```
# define a function to calculate custom weights
def my_weight_func(distances):
    # create an array of ones with the same shape as distances
    weights = [1] * len(distances)
    # find the index of the closest neighbor
    closest_idx = distances.argmin()
    # assign a weight of 1.1 to the closest neighbor
    weights[closest_idx] = 1.1
    return weights
```

Σχήμα 4.7: Μέθοδος Ανάθεσης Βαρών

4.4.2 Adann NR One Pass

Η επόμενη υλοποίηση που θα αναλυθεί είναι η παραλλαγή AdaNN NR One Pass. Συγκεκριμένα και σε αυτήν και στην επόμενη παραλλαγή την AdaNN NR Multipass θα πραγματοποιηθεί αφαίρεση του θορύβου. Το σημαντικό κομμάτι το οποίο θα πρέπει να αναλυθεί είναι σε ποιο σημείο θα γίνει η αφαίρεση του θορύβου. Αυτό το οποίο έχει υλοποιηθεί είναι κατά την φάση του υπολογισμού του βέλτιστου κ για κάθε ένα στιγμιότυπο όταν γίνονται διαδοχικές εκτελέσεις του αλγορίθμου KNN αυξάνεται το κ κατά 1 κάθε φορά που φτάνει το μη ανεκτό όριο για το βέλτιστο κ. Στην προηγούμενη παραλλαγή του Adann χωρίς αφαίρεση θορύβου χρησιμοποιήθηκαν διάφορες συναρτήσεις για τον υπολογισμού του ορίου του βέλτιστου κ. Στις παραλλαγές της αφαίρεσης θορύβου χρησιμοποιείται σαν όριο η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων της κλάσης στην οποία ανήκει. Αυτό σημαίνει πως σε αυτή την παραλλαγή αλλάζει η συνάρτηση του υπολογισμού του ορίου και μέσα στον κώδικα.

```
def calculateMaximumKLimit(self, y_test, y_train):

    testingInstanceClass =  y_test[0]
    return round(np.sqrt(y_train.count(testingInstanceClass) + 1))
```

Σχήμα 4.8: Μέθοδος Υπολογισμού Ορίου Βέλτιστου Κ

Ένα άλλο σημαντικό σημείο το οποίο πρέπει να αναφερθεί είναι η τροποποίηση που έχει η μέθοδος calculateBestK η οποία εν τέλει θα επιστρέψει την τιμή του βέλτιστου για το δοθέν στιγμιότυπο. Σε αυτό το σημείο είναι σημαντική η διαφορά καθώς όταν η τιμή του κ φτάνει το μέγιστο επιτρεπόμενο όριο, δεν θα γίνει επιστροφή της τιμής του κ όπως στην πρώτη παραλλαγή, αλλά θα γίνει επιστροφή της τιμής -1 για να δείξει στον αλγόριθμο πως το στιγμιότυπο αυτό είναι θόρυβος.

```
def calculateBestK(self, x_train, y_train, x_test, y_test):
    y_test = [y_test]
    x_test = x_test.reshape(1,-1)

    numberOfTrainingInstances = len(x_train)

    k_limit = self.calculateMaximumKLimit(y_test, y_train)

    #print('\t \t k limit = ' + str(k_limit))

    k = 1
    while True:

        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(x_train, y_train)
        y_pred = knn.predict(x_test)

        if metrics.accuracy_score(y_test, y_pred) == 1.0 :
             break

        if k == self.calculateMaximumKLimit(y_test, y_train):
            return -1
            break
        k+=1
    return k
```

Σχήμα 4.9: Μέθοδος Υπολογισμού Βέλτιστου Κ

Εφόσον ο αλγόριθμος έχει γνώση για το αν ένα στιγμιότυπο αποτελεί θόρυβο ή όχι πλέον έρχεται η ώρα της διαλογής. Όπως και στην προηγούμενη παραλλαγή υπάρχει διάβασμα του αρχείου και στην συνέχεια γίνεται υπολογισμός του κάθε ένα βέλτιστου κ για κάθε ένα στιγμιότυπο αλλά η διαφοροποίηση θα γίνει στην στιγμή όπου θα γίνει μια ακόμα επανάληψη πάνω στα στιγμιότυπα και θα γίνει έλεγχος για κάθε ένα στιγμιότυπο για το αν το κ το οποίο θα έχει είναι αρνητικό ή όχι. Οπότε οι περιπτώσεις στις οποίες τα στιγμιότυπα είναι θόρυβος θα έχουν αρνητικό κ και συνεπώς δεν θα συμπεριληφθούν στο σύνολο των δεδομένων. Επίσης κάτι άλλο το οποίο θα παρατηρηθεί στην εικόνα είναι πως κατά την διάρκεια αυτής της διαδικασίας γίνεται υπολογισμός των χαρακτηριστικών που επισημάνθηκαν σαν θόρυβος και αφαιρέθηκαν καθώς και επιστρέφεται και ο συνολικός αριθμός των στιγμιοτύπων για στατιστικούς λόγους.

```
#convert list to array
k_array = np.array(k_list)

#add the collumn of the k to the data
data_with_k = np.column_stack((data, k_array))


#for every instance we check if k > 0 (not marked as noise)
#and we write the line to the new k_trainDatasetFileName file
#line format: x1 x2 x3 x4 y k
i = 0
while i < data_with_k.shape[0]:

    x_train = list(x)[:i] + list(x)[i+1:]
    x_test = x[i]
    y_train = list(y)[:i] + list(y)[i+1:]
    y_test = y[i]

    if (k_array[i] > 0):

        newLine = ""
        for j in range(x.shape[1]):
            newLine += str(x_test[j]) + dSep
        newLine += str(int(y[i])) + dSep + str(k_array[i]) + "\n"
        datasetWithK.write(newLine)
    else:
        instancesMarkedAsNoise = instancesMarkedAsNoise + 1

    i=i+1


datasetWithK.close()
print('\t Done' + folder + '\n')
get_ipython().magic('reset -sf')

noise_removal_details = (instancesMarkedAsNoise, totalInstances)
return noise_removal_details
```

Σχήμα 4.10: Μέθοδος Υπολογισμού Βέλτιστου Κ

4.4.3 Adann NR MultiPass

Η τελευταία υλοποίηση η οποία θα αναλυθεί είναι AdaNN NR Multipass η οποία είναι άλλη μια παραλλαγή του AdaNN για αφαίρεση θορύβου. Η σημαντική διαφοροποίηση του AdaNN NR Multipass είναι ότι σε αντίθεση με την προηγούμενη παραλλαγή αυτή θα κάνει πολλαπλά περάσματα στα δεδομένα του συνόλου εκπαίδευσης προκειμένου να μην υπάρχει κανένα στιγμιότυπο το οποίο να θεωρείται θόρυβος. Για να γίνει αυτό σημαίνει ότι κάθε στιγμιότυπο του συνόλου εκπαίδευσης θα πρέπει να έχει ένα βέλτιστο κ το οποίο να μην προσπερνάει το μέγιστο όριο το οποίο χρησιμοποιούμε. Το μέγιστο όριο το οποίο χρησιμοποιείται και σε αυτήν την παραλλαγή είναι η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων της κλάσης στην οποία ανήκει το στιγμιότυπο.
Όπως φαίνεται στην παρακάτω εικόνα 4.11 πρώτα από όλα αφότου έχει γίνει διάβασμα του αρχείου αρχικοποιούνται 2 μεταβλητές, η πρώτη μεταβλητή totalInstances θα περιέχει τον αριθμο των στιγμιοτύπων και η δεύτερη θα περιέχει τον αριθμό των στιγμιοτύπων που θα κατηγοριοποιηθούν σαν θόρυβος. Από εκεί και πέρα για όσο υπάρχει θόρυβος θα εκτελείται. Αρχικά θα αρχικοποιηθεί μια κενή λίστα με τα κ τα οποία θα υπολογιστούν. Για αρχή με μια δομή επανάληψης θα υπολογιστεί το κ για κάθε στιγμιότυπο του συνόλου εκπαίδευσης και θα γε-

μίσει αυτή η λίστα k_list. Στην συνέχεια πάλι με μια δομή επανάληψης θα γίνει προσπέλαση όλων των στιγμιοτύπων και θα αφαιρεθούν εκείνα τα στιγμιότυπα τα οποία έχουν αρνητικό κ. Σε περίπτωση που που βρεθούν στιγμιότυπα με κ αρνητικο σημαίνει ότι υπάρχει θόρυβος στο ακόμα στα δεδομένα μας και θα συνεχίσει να εκτελείται η while(noiseExists) αλλιώς θα σταματήσει.

```
datasetWithK = open(folder + str("k_") + trainingDataSetFileName, "a")


totalInstances = x.shape[0]
instancesMarkedAsNoise = 0

noiseExists = True
k_array = []
while noiseExists:

    i = 0
    k_list = list()
    # We calculate k for each instance of training data
    while i < x.shape[0]: #number of instances
        x_train = list(x)[:i] + list(x)[i+1:]
        x_test = x[i]
        y_train = list(y)[:i] + list(y)[i+1:]
        y_test = y[i]

        k = self.calculateBestK(x_train, y_train, x_test, y_test)
        k_list.append(k)
        i = i + 1


    i = 0
    noiseInstancesFound = 0
    #
    while i < x.shape[0]: #number of instances
        x_train = list(x)[:i] + list(x)[i+1:]
        x_test = x[i]
        y_train = list(y)[:i] + list(y)[i+1:]
        y_test = y[i]

        if (k_list[i] < 0):
            noiseInstancesFound = noiseInstancesFound + 1

            x = np.delete(x, i, 0) # delete the i element of 0 axis (rows)
            y = np.delete(y, i, 0)  # delete the i element of 0 axis (rows)
            k_list = np.delete(k_list, i, 0)  # delete the i element of 0 axis (rows)
            instancesMarkedAsNoise = instancesMarkedAsNoise + 1
            i = i - 1

        i = i + 1

    if (noiseInstancesFound == 0):
        noiseExists = False
        k_array = k_list
    else:
        noiseExists = True
```

Σχήμα 4.11: Μέθοδος Υπολογισμού βέλτιστων κ και αφαίρεσης θορύβου Μέρος (α)

Αμέσως μετά εφόσον γίνει αυτή η αφαίρεση των στιγμιοτύπων τα οποία θεωρούνται θόρυβος έρχεται η ώρα για να γραφτούν τα στιγμιότυπα του συνόλου εκπαίδευσης 4.12 αλλά μαζί πλέον με τα βέλτιστα κ τους. Για να γίνει αυτό έχει υλοποιηθεί μια δομή επανάληψης η οποία θα γράψει κάθε μια γραμμή στο αρχείο αφού κάνει έναν ακόμα έλεγχο για θετικό κ κατά την προσπέλαση των στιγμιοτύπων. Αυτή η μέθοδος που αναλύεται εδώ είναι η οποία διαφοροποιείται από την παραλλαγή AdaNN NR One Pass και από τις απλές παραλλαγές μεταβλητό όριο. Η διαφοροποίηση αυτής της μεθόδου έχει να κάνει με τον τρόπο με τον οποίο αφαιρεί τον θόρυβο ο αλγόριθμος κατά την φάση του υπολογισμού των βέλτιστων κ. Κατά την φάση της εξαγωγής συμπεράσματος γίνεται όπως και στις προηγούμενες παραλλαγές εύρεση του εγγύτερου στιγμιότυπου από το σύνολο εκπαίδευσης και το εξεταζόμενο στιγμιότυπο του συνόλου ελέγχου θα πάρει το βέλτιστο κ του πρώτου.

```
    # We check every k calculated
    # and if it is not marked as noise we write it to the file
    i = 0
    while i < x.shape[0]:

        x_train = list(x)[:i] + list(x)[i+1:]
        x_test = x[i]
        y_train = list(y)[:i] + list(y)[i+1:]
        y_test = y[i]

        if (k_array[i] > 0):

            newLine = ""
            for j in range(x.shape[1]):
                newLine += str(x_test[j]) + dSep
            newLine += str(int(y[i])) + dSep + str(k_array[i]) + "\n"
            datasetWithK.write(newLine)
        i=i+1


    datasetWithK.close()

    get_ipython().magic('reset -sf')

    print('\t Done' + folder + '\n')

    noise_removal_details = (instancesMarkedAsNoise, totalInstances)
    return noise_removal_details
```

Σχήμα 4.12: Μέθοδος Υπολογισμού βέλτιστων κ και αφαίρεσης θορύβου Μέρος (β)

Κεφάλαιο 5

Πειραματική μελέτη

5.1 Σύνολα δεδομένων

Το πρώτο σημείο στο οποίο είναι σημαντικό για την πειραματική μελέτη των παραλλαγών είναι να γίνει αναφορά στα σύνολα δεδομένων που χρησιμοποιήθηκαν [22]. Το πρώτο σύνολο δεδομένων που χρησιμοποιήθηκε είναι το iris το οποίο είναι ένα κλασικό σύνολο δεδομένων από τα πρώτα τα οποία χρησιμοποιήθηκαν για κατηγοριοποίηση και για έλεγχο μεθόδων κατηγοριοποίησης. Συγκεκριμένα αυτό το σύνολο δεδομένων έχει δεδομένα από τον τομέα του Life Science με την μορφή πίνακα και αποτελείται από 150 στιγμιότυπα. Επίσης σημαντικό είναι ότι έχει 4 χαρακτηριστικά τα οποία είναι πραγματικές τιμές. Επίσης σημαντικό είναι ότι περιλαμβάνει σαν στόχους 3 κλάσεις όπου κάθε μια από αυτές έχει από 50 στιγμιότυπα τα οποία είναι στιγμιότυπα από το φυτό iris και περιγράφει τις κλάσεις Iris Setosa, Iris Versicolour και Iris Virginica.

Το δεύτερο σύνολο δεδομένων που θα περιγραφεί είναι το mgt (Magic Gamma Telescope). Τα δεδομένα είναι καταγραφές από προσομοίωση από σωματίδια γάμμα υψηλής ενέργειας σε ένα τηλεσκόπιο. Το διάσημο αυτό τηλεσκόπιο ονομάζεται Cherenkov και καταγράφει ακτίνες γάμμα χάρη στην ακτινοβολία που εκπέμπουν τα σωματίδιά. Αυτό το σύνολο των δεδομένων ανήκει στον χώρο της φυσικής και αποτελείται από 19020 στιγμιότυπα. Συγκεκριμένα τα χαρακτηριστικά τους είναι 10. Η μια παραπάνω στήλη η οποία θα βρεθεί είναι η κατηγορία που ανήκει το κάθε στιγμιότυπο. Σημαντικό είναι πως το κάθε χαρακτηριστικό έχει πραγματικές τιμές. Συνολικά έχει 2 κλάσεις.

Το τρίτο σύνολο που θα περιγραφεί είναι το Phoneme. Ο στόχος αυτού του συνόλου δεδομένων είναι να γίνει διαχωρισμός των ρινικών και των στοματικών ήχων. Το σύνολο των δεδομένων αποτελείται από 3818 στιγμιότυπα με ένρινους ήχους και 1586 στιγμιότυπα από στοματικούς ήχους, αριθμώντας συνολικά 5404 στιγμιότυπα. Το συγκεκριμένο σύνολο δεδομένων περιλαμβάνει 5 χαρακτηριστικά τα οποία είναι πραγματικοί αριθμοί. Επίσης σημαντικό είναι ότι τα δεδομένα δεν έχουνε ελλείπουσες τιμές. Συνολικά έχει 2 κλάσεις.

Το τέταρτο σύνολο δεδομένων που θα περιγραφεί είναι το Landsat Satellite το οποίο αποτελείται από 6435 στιγμιότυπα και είναι ένα σύνολο δεδομένων που ανήκει στις φυσικές επιστήμες. Συγκεκριμένα αποτελείται από 36 χαρακτηριστικά τα οποία είναι όλα ακέραιοι αριθμοί και ανήκουν όλα στο εύρος 0-255. Αυτό το σύνολο δεδομένων περιλαμβάνει τις τιμές από pixel από πολλαπλά φάσματα από περιοχές μεγέθους 3x3 από δορυφορική φωτογραφία. Συνολικά έχει 6 κλάσεις.

Το πέμπτο σύνολο δεδομένων που θα περιγραφεί είναι το Banana. Συγκεκριμένα το σύνολο αυτό περιλαμβάνει 5300 στιγμιότυπα και δεν έχει καθόλου ελλείπουσες τιμές. Συγκεκριμένα αυτό το σύνολο δεδομένων περιλαμβάνει 2 χαρακτηριστικά τα οποία είναι είναι πραγματικοί αριθμοί. Το 2 χαρακτηριστικά At1 και At2 που αντιστοιχούν στον άξονα και x και στον άξονα y αντίστοιχα. Η ετικέτα της κλάσης -1 και 1 αντιπροσωπεύουν ένα διαφορετικό σχήμα μπανάνας.

Το έκτο σύνολο δεδομένων που θα περιγραφεί είναι το Texture. Συγκεκριμένα το σύνολο αυτό περιλαμβάνει 5500 στιγμιότυπα και δεν έχει καθόλου ελλείπουσες τιμές. Αυτό το σύνολο δεδομένων περιλαμβάνει 40 χαρακτηριστικά τα οποία είναι είναι πραγματικοί αριθμοί. Σημαντικό είναι πως το συγκεκριμένο σύνολο δεδομένων έχει 11 κατηγορίες στις οποίες μπορεί να κατηγοριοποιηθεί το στιγμιότυπο.

Το έβδομο σύνολο δεδομένων που θα περιγραφεί είναι το Pima Indians Diabetes dataset. Συγκεκριμένα το σύνολο αυτό περιλαμβάνει 768 στιγμιότυπα. Αναφορά επίσης πρέπει να γίνει στο ότι αποτελείται από 8 χαρακτηριστικά τα οποία είναι πραγματικοί αριθμοί. Επίσης πολύ σημαντικό είναι πως το συγκεκριμένο dataset δεν έχει καθόλου τιμές που λείπουν. Το συγκεκριμένο dataset προέρχεται από το Εθνικό Ινστιτούτο Διαβήτη και Πεπτικών Νεφροπαθειών και κάθε ένα στιγμιότυπο είναι ένας ασθενής της Ινδιάνικης φυλής Pima. Η κατηγορία η οποία θα πρέπει να διαπιστωθεί είναι άμα ο ασθενής είναι αρνητικά ή θετικά διαγνωσμένος στον διαβήτη.

Το όγδοο dataset που θα περιγραφεί είναι το Yeast το οποίο προβλέπει την τοποθεσία της κυτταρικής εντόπιση πρωτεϊνών. Αυτό το dataset κυρίως χρησιμοποιείται για κατηγοριοποιήσεις και αποτελείται από 8 χαρακτηριστικά τα οποία είναι πραγματικοί αριθμοί. Όσον αφορά το μέγεθος του dataset, αποτελείται από 1484 στιγμιότυπα. Επίσης σημαντικό αναφοράς είναι ότι σε αυτό το dataset υπάρχουν 10 κλάσεις για τις πιθανές αυτές τοποθεσίες.

Το ένατο dataset το οποίο θα περιγραφεί είναι Balance Scale Dataset το οποίο περιγράφει αποτελέσματα ψυχολογικών πειραμάτων. Σημαντικό είναι ότι αυτό το dataset αποτελείται από 625 στιγμιότυπα και το κάθε ένα από αυτά έχει 4 χαρακτηριστικά τα οποία είναι πραγματικοί αριθμοί. Επίσης σημαντικό είναι ότι κάθε στιγμιότυπο μπορεί να κατηγοριοποιηθεί σε 3 κλάσεις αν έχει ισορροπία, κλίση προς τα αριστερά ή κλίση προς τα δεξιά.

Το δέκατο dataset είναι το Letter Recognition το οποίο είναι ένα dataset το οποίο αποτελείται από 20000 στιγμιότυπα και 26 κλάσεις. Συγκεκριμένα κάθε μια κλάση είναι ένα γράμμα της αγγλικής αλφαβήτου. Κάθε ένα στιγμιότυπο συγκεκριμένα αποτελείται από 16 χαρακτηριστικά τα οποία είναι όλα ακέραιοι αριθμοί.

Το ενδέκατο dataset το οποίο χρησιμοποιήθηκε είναι το Ecoli. Το dataset αυτό αποτελείται από 336 στιγμιότυπα από τα οποία κάθε ένα αποτελείται από 7 χαρακτηριστικά τα οποία είναι πραγματικοί αριθμοί. Ο στόχος που υπάρχει για επίλυση σε αυτό το dataset είναι να βρεθεί η τοποθεσία κάποιων πρωτεϊνών με βάση κάποια χαρακτηριστικά του κυττάρου Escherichia coli. Οι κλάσεις οι οποίες υπάρχουν σε αυτό το πρόβλημα είναι οι 8 διαφορετικές πιθανές τοποθεσίες.

Το δωδέκατο dataset το οποίο χρησιμοποιήθηκε είναι το Pen Digits data set το οποίο είναι ένα σύνολο το οποίο περιλαμβάνει αναγνωρίσεις ψηφίων οι οποίες είναι γραμμένες από μια πένα. Το συγκεκριμένο dataset αποτελείται από 10992 στιγμιότυπα από τα οποία το κάθε ένα αποτελείται από 16 χαρακτηριστικά τα οποία είναι ακέραιοι αριθμοί. Συγκεκριμένα τα στιγμιότυπα αυτά είναι από 44 συγγραφείς όπου έγινε λήψη των ψηφίων αυτών δειγματοληπτικά. Ο αριθμός των διαφορετικών κατηγοριών που μπορεί να προκύψουν σε αυτό το πρόβλημα είναι 10, όσοι δηλαδή οι διαφορετικοί αριθμοί.

Το δέκατο τρίτο dataset το οποίο χρησιμοποιήθηκε είναι το Twonorm dataset το οποίο αποτελείται συνολικά από 7400 στιγμιότυπα. Συγκεκριμένα κάθε στιγμιότυπο αποτελείται από 20 χαρακτηριστικά τα οποία είναι όλα πραγματικοί αριθμοί. Ο αριθμός των κλάσεων που μπορούν να προκύψουν σαν αποτέλεσμα σε αυτό το πρόβλημα είναι 2.

Πίνακας 5.1: Dataset description

Dataset	Size	Attributes	Classes
Balance (bl)	625	4	3
Banana (bn)	5300	2	2
Ecoli (ecl)	336	7	8
Iris	150	4	3
Letter Recognition (lir)	20000	16	26
Landsat Satellite (ls)	6435	36	6
Magic G. Telescope (mgt)	19020	10	2
Pen-Digits (pd)	10992	16	10
Phoneme (ph)	5404	5	2
Pima (pm)	768	8	2
Twonorm (tn)	7400	20	2
Texture (txr)	5500	40	11
Yeast (ys)	1484	8	10

5.2 Εγκαθίδρυση πειραμάτων

Σε αυτό το σημείο είναι ιδιαίτερα σημαντικό να σταθούμε πως χρησιμοποιήθηκαν αυτές οι μέθοδοι οι οποίες υλοποιήθηκαν για κάθε μια παραλλαγή που έχει αναλυθεί στις προηγούμενες ενότητες. Προκειμένου να παραχθούν τα τελικά αρχεία excel με οργανωμένα τα αποτελέσματα έγινε μια υλοποίηση σε κώδικα η οποία θα μπορεί να τρέξει μαζικά τις παραλλαγές που υπάρχουν έτσι ώστε να είναι συγκρίσιμες μεταξύ τους. Το πρώτο κομμάτι κώδικα 5.1 θα ορίσει μια κλάση η οποία θα ορίζει την έξοδο του standard output της κονσόλας της Python να γράφονται στο αρχείο MYLOGS.txt έτσι κατά την διάρκεια της εκτέλεσης των πειραμάτων να είναι κατανοητό σε ποια φάση βρίσκεται η εκτέλεση τους. Σημαντικό να αναφερθεί είναι πως για όλα τα dataset που θα αναφερθούν παρακάτω δημιουργήθηκαν άλλες 2 παραλλαγές αυτών που έχουν το επίθεμα 10 και 30. Αυτές οι παραλλαγές των dataset σημαίνουν πως τροποποήθηκε η κλάση του κάθε ενός στιγμιοτύπου του dataset με πιθανότητα 10 ή 30 να ταξινομηθούν λάθος.

Το επόμενο κομμάτι κώδικα το οποίο θα αναλυθεί περιλαμβάνει κάποιες γραμμές που χρειάζονται για τις γενικές ρυθμίσεις των πειραμάτων που θα εκτελεστούν. Αρχικά θα οριστεί η σχετική διαδρομή για τους φακέλους που θα έχουν τα σύνολα ελέγχου καθώς και τα σύνολα εκπαίδευσης. Στην συνέχεια θα γίνει άνοιγμα ενός αρχείου Excel με όνομα AdaNN.xlsx στο οποίο θα ορίσουμε ένα νέο φύλλο εργασίας. Επίσης σε αυτό το φύλλο γράφουμε τους τίτλους κάθε στήλης που θα εμφανίζεται στο αρχείο. Στην πρώτη θα εμφανίζεται το όνομα του συνόλου δεδομένων, στην δεύτερη στήλη θα εμφανίζεται το όνομα της συνάρτησης με την οποία θα υπολογίζεται το μέγιστο όριο του βέλτιστου κ και στις επόμενες στήλες θα είναι η τιμή της κάθε μετρικής που θέλουμε να εξετάσουμε. Η πρώτη μετρική είναι το Recall, η δεύτερη μετρική είναι το Precision, η τρίτη μετρική είναι το Fscore και τέλος η πιο σημαντική μετρική που είναι η ακρίβεια. Παρακάτω βλέπουμε πως αρχικοποιούμε έναν πίνακα με τα k για κάθε σύνολο δεδομένων με τα οποίο γίνεται η βέλτιστη κατηγοριοποίηση σύμφωνα με το άρθρο [23] προκει-

```python
from AdaNN import *
from ApplyAdaNN import *
from KNN import predict_and_evaluate
import os
import xlsxwriter
import sys

#Set a logger
class Logger(object):
    def __init__(self, filename="Default.log"):
        self.terminal = sys.stdout
        self.log = open(filename, "a")

    def write(self, message):
        self.terminal.write(message)
        self.log.write(message)
        self.log.flush()

    def flush(self):
        pass

sys.stdout = Logger("MYLOGS.txt")
```

Σχήμα 5.1: Κώδικας δημιουργίας αρχείου καταγραφής

μένου να γίνει μια καλύτερη σύγκριση των αποτελεσμάτων των παραλλαγών με τα καλύτερά δυνάτα.

```python
path_to_datasets = './Datasets/data'

workbook = xlsxwriter.Workbook('AdaNN.xlsx')
worksheet = workbook.add_worksheet()

dataset_folders = os.listdir(path_to_datasets)

#Write headers of file

worksheet.write(0, 0, 'Dataset')
worksheet.write(0, 1, 'Limit Function')

worksheet.write(0, 3, 'Avg Recall')
worksheet.write(0, 4, 'Avg Precision')
worksheet.write(0, 5, 'Avg Fscore')
worksheet.write(0, 6, 'Avg Support')
worksheet.write(0, 7, 'Accuracy')

#Array of best k for each dataset
k_array = {
"bl": 42, "bl10": 43, "bl30": 32, "bn": 29, "ecl": 6, "ecl10": 6,
"ecl30": 14, "iris": 11, "lir": 4, "ls": 8, "ls10": 8, "ls30": 13,
"mgt": 10, "mgt10": 20, "pd": 1, "pd10": 8, "pd30": 12, "ph": 1,
"ph10": 8, "ph30": 48, "pm": 48, "pm10": 18, "pm30": 47, "sh": 1,
"tn": 49, "tn10": 47, "tn30": 50, "txr": 1, "txr10": 5,"txr30": 10,
"ys": 14, "ys10": 13, "ys30": 14,
```

Σχήμα 5.2: Αρχείο Testing μέρος (α)

Για αρχή έχει γίνει αποθήκευση σε μια μεταβλητή η διαδρομή στην οποία είναι αποθηκευμένα τα σύνολα δεδομένων και αποθηκεύεται σε μια μεταβλητή dataset_folders μια λίστα με τα ονόματα των αρχείων που είναι μέσα στο φάκελο. Αμέσως μετά αρχικοποιούμε και το αρχείο excel το οποίο θα έχει το σύνολο των αποτελεσμάτων και στην συνέχεια θα τρέξει ο αλγόριθμος και να γίνει εξαγωγή συμπεράσματος για κάθε ένα από αυτά. Για αρχή κάνουμε μια αρχικοποίηση των διαφορετικών συναρτήσεων ορίου του βέλτιστου κ. Αυτές είναι η τετραγωνική ρίζα, η κυβική ρίζα, η σταθερή συνάρτηση η οποία είναι η fixed τιμή 9 ως μέγιστο όριο,

καθώς και η τετραγωνική ρίζα του αριθμού των στιγμιοτύπων της κλάσης του εξεταζόμενου στιγμιότυπου. Στην συνέχεια θα γίνει εκτέλεση για κάθε κάθε φάκελο - dataset θα τρέξουν όλες οι συναρτήσεις . Για αρχή θα φτιαχτεί ένα αντικείμενο της κλάσης AdaNN όπου θα καλέσουμε την μέθοδο που θα δημιουργεί τα αρχεία τα οποία είναι της μορφής datasetname_k και μετά στην συνέχεια με την άλλη μέθοδο method1 θα γίνει εξαγωγή του αποτελέσματος για εκείνο το σύνολο δεδομένων και την δεδομένη συνάρτηση. Τέλος αφότου γίνει υπολογισμός των μετρικών και εμφάνιση τους στην κονσόλα θα γίνει εγγραφή των αποτελεσμάτων δύο γραμμές παρακάτω από την τελευταία εγγραφή.

```
available_functions = ['sqrt', 'cbrt', 'const', 'sqrt_limit_class']
excelRowIndex = 0


for folder in dataset_folders:


    for func in available_functions:

        excelRowIndex = excelRowIndex + 2 #

        print('Building model')


        model = AdaNN(func)

        print('Working on dataset...' + folder)

        model.method(path_to_datasets  + '/' + folder + '/',  'data1', '    ')

        print('Applying on dataset...' + folder)

        scores_tuple  = method1(path_to_datasets + '/' + folder + '/',  'data_te1', 'k_data1',' ')

        precision, recall, fscore, supp, accuracy = scores_tuple


        print("---------------Algorithm Results--------------------------")
        print(scores_tuple)

        #---------------Algorithm Results--------------------------

        print('Writing results to excel file...')


        worksheet.write(excelRowIndex, 0, str(folder))
        worksheet.write(excelRowIndex, 1, str(func))


        worksheet.write(excelRowIndex, 3, str(recall))
        worksheet.write(excelRowIndex, 4, str(precision))
        worksheet.write(excelRowIndex, 5, str(fscore))
        worksheet.write(excelRowIndex, 6, str(supp))
        worksheet.write(excelRowIndex, 7, str(accuracy))


        os.remove(path_to_datasets  + '/' + folder + '/k_data1')
```

Σχήμα 5.3: Αρχείο Testing μέρος (β)

Τέλος για την εγγραφή του Best KNN έχει γραφτεί το παραπάνω κομμάτι κώδικα 5.3 το οποίο θα κάνει εκτέλεση των διαφορετικών KNN για κάθε σύνολο δεδομένων που υπάρχει στον φάκελο καθώς και για κάθε ένα σύνολο δεδομένων θα γίνει υπολογισμός των βασικών μετρικών για την βέλτιστη τιμή K όπως έχει οριστεί στον πίνακά k_array όπως αναφέρθηκε στην φωτογραφία παραπάνω. Ιδιαίτερα σημαντικό να αναφερθεί είναι πως το αρχείο Testing το οποίο έχει αναλυθεί περιλαμβάνει κομμάτια κώδικα για την πρώτη παραλλαγή στην οποία γίνονται διαδοχικές δοκιμές για κάθε συνδυασμό συνάρτησης μέγιστου ορίου για το βέλτιστο κ αλλά χωρίς ακόμα να γίνεται αναφορά στον τρόπο με τον οποίο θα γίνει εκτέλεση των πειραμάτων της αφαίρεσης του θορύβου στις παραλλαγές του AdaNN NR One Pass και AdaNN NR Multiple Pass. Επίσης σε αυτό το σημείο αυτός ο κώδικας 5.4 χρησίμευσε για την παραγωγή

των μετρικών για αξιολόγησης για τις τιμές 1-KNN, 10-KNN και Best-KNN για κάθε σύνολο δεδομένων.

```
        # print("-------------------------------------------------------------")


        print("---------------KNN-----------------------------------------")


        training = path_to_datasets  + '/' + folder + '/data1'
        testing = path_to_datasets + '/' + folder + '/data_te1'

        scores_tuple  = predict_and_evaluate(training, testing, k_array[folder])

        print(scores_tuple)

        precision, recall, fscore, supp, accuracy = scores_tuple

        excelRowIndex = excelRowIndex + 2 #

        worksheet.write(excelRowIndex, 0, str(folder))
        worksheet.write(excelRowIndex, 1, str('KNN'))


        worksheet.write(excelRowIndex, 3, str(recall))
        worksheet.write(excelRowIndex, 4, str(precision))
        worksheet.write(excelRowIndex, 5, str(fscore))
        worksheet.write(excelRowIndex, 6, str(supp))
        worksheet.write(excelRowIndex, 7, str(accuracy))

        print("-------------------------------------------------------------")



    workbook.close()
```

Σχήμα 5.4: Αρχείο Testing μέρος (β)

Ιδιαίτερα σημαντική αναφορά πρέπει να γίνει στην διαφοροποίηση του Testing αρχείου στις παραλλαγές όπου γίνεται η αφαίρεση θορύβου. Όπως φαίνεται στην παρακάτω φωτογραφία 5.5 βλέπουμε ότι γίνεται η ίδια διαδικασία όσον αφορά το διάβασμα των αρχείων και όσον αφορά την εφαρμογή του μοντέλου και όσον αφορά την εγγραφή των αποτελεσμάτων στο αρχείο που έχει τα τελικά αποτελέσματα. Αυτό το οποίο διαφοροποιεί αυτήν την παραλλαγή είναι ότι η μέθοδος που κάνει τον υπολογισμό των βέλτιστων κ για κάθε ένα στιγμιότυπο θα επιστρέψει ένα noise_removal_tuple το οποίο θα περιλαμβάνει τον αριθμό των στιγμιοτύπων τα οποία χαρακτηρίστηκαν ως θόρυβος καθώς και τον αριθμό των στιγμιοτύπων που είχαν αρχικά το σύνολο των δεδομένων. Αυτό το οποίο προκύπτει είναι ότι θα υπάρχουν 3 παραπάνω στήλες στο αρχείο οι οποίες θα έχουν τον αριθμό των στιγμιοτύπων που είναι θόρυβος, το αρχικό μέγεθος του συνόλου δεδομένων καθώς και ένα ποσοστό το οποίο δείχνει το θόρυβο επί του συνολικού μεγέθους του συνόλου δεδομένων.

```
available_functions = ['sqrt_limit_class']
excelRowIndex = 0

for folder in dataset_folders:

    for func in available_functions:

        excelRowIndex = excelRowIndex + 2 #

        print('Building model')


        model = AdaNN(func)

        print('Working on dataset...' + folder)

        noise_removal_tuple = model.method(path_to_datasets  + '/' + folder + '/',  'data1', '  ')

        print('Applying on dataset...' + folder)

        scores_tuple  =  method1(path_to_datasets + '/' + folder + '/',  'data_te1', 'k_data1','    ')

        precision, recall, fscore, supp, accuracy = scores_tuple
        instancesMarkedAsNoise, totalInstances = noise_removal_tuple


        print("--------------Algorithm Results---------------------------")
        print(scores_tuple)

        #--------------Algorithm Results--------------------------

        print('Writing results to excel file...')


        worksheet.write(excelRowIndex, 0, str(folder))
        worksheet.write(excelRowIndex, 1, str(func))


        worksheet.write(excelRowIndex, 3, str(recall))
        worksheet.write(excelRowIndex, 4, str(precision))
        worksheet.write(excelRowIndex, 5, str(fscore))
        worksheet.write(excelRowIndex, 6, str(supp))
        worksheet.write(excelRowIndex, 7, str(accuracy))

        worksheet.write(excelRowIndex, 8, str(instancesMarkedAsNoise))
        worksheet.write(excelRowIndex, 9, str(totalInstances))
        worksheet.write(excelRowIndex, 10, str(instancesMarkedAsNoise / totalInstances * 100))


        #os.remove(path_to_datasets  + '/' + folder + '/k_data1')

    print("------------------------------------------------------------")
```

Σχήμα 5.5: Αρχείο Testing μέρος (β)

5.3 Πειραματικές μετρήσεις

Σε αυτό το σημείο είναι σημαντικό να γίνει αναφορά στα αποτελέσματα τα οποία εξάχθηκαν από την εκτέλεση του κώδικα ο οποίος έχει σχολιαστεί. Στον παρακάτω πίνακα 5.2 βλέπουμε την παράθεση όλων των αποτελεσμάτων. Στον πίνακα φαίνεται για κάθε σύνολο δεδομένων και για κάθε παραλλαγή το αποτέλεσμα της ακρίβειας των αποτελεσμάτων. Βλέπουμε πως η πρώτη στήλη είναι το σύνολο δεδομένων όπου έχουμε τα 32 αρχεία τα οποία αποτελούν σύνολα δεδομένων. Ιδιαίτερα σημαντική λεπτομέρεια είναι ότι τα σύνολα δεδομένων τα οποία έχουν το επίθεμα 10 και 30 περιλαμβάνουν ένα ποσοστό θορύβου για να γίνει αντιληπτή η απόδοση του αλγόρίθμου και τι αντοχή έχει στον θόρυβο. Αυτός ο θόρυβος είναι τοποθετημένος μέσα στα σύνολα δεδομένων ώστε να γίνει αντιληπτή η απόδοση του αλγορίθμου και τι αντοχή έχει σε θόρυβο. Στην δεύτερη στήλη υπάρχει η στατική εκτέλεση ενός αλγορίθμου KNN με αριθμό εγγύτερων γειτόνων να ισούται με 1. Το ίδιο συμβαίνει και στην τρίτη στήλη όπου ο αριθμός των εγγύτερων γειτόνων ισούται με 10 για την εξαγωγή συμπεράσματος για τα στιγμιότυπα του συνόλου ελέγχου.Αυτό γίνεται προκειμένου να γίνει η σύγκριση με τις άλλες παραλλαγές. Η τέταρτη στήλη είναι το κ για το πετυχαίνει την μεγαλύτερη ακρίβεια ο αλγόριθμος KNN σύμφωνα με τα κ όπως αναφέρονται στο άρθρο [23]. Πολύ σημαντικό να αναφερθεί είναι πως κάποιες τιμές διαφέρουν ελάχιστα σε σχέση με τον παρακάτω πίνακα και τις τιμές του άρθρου, πιθανών λόγω της διαφοράς στην υλοποίηση του αλγορίθμου KNN. Στην παρούσα διπλωματική η εκδοχή του KNN που χρησιμοποιήθηκε είναι βιβλιοθήκη του scikit learn. Η πέμπτη στήλη με όνομα const ουσιαστικά είναι το αποτέλεσμα της παραλλαγής του αλγορίθμου του AdaNN όπου το μέγιστο όριο της τιμής του βέλτιστου κ ισούται με την τιμή 9. Αυτή είναι η δεύτερη στήλη με την οποία θα γίνεται σύγκριση. Οι δύο στήλες με τις οποίες θα συγκριθούν οι παραλλαγές είναι το Best KNN και const.

Το πρώτο σημείο το οποίο παρατηρούμε είναι ότι η const είναι καλύτερη από την sqrt παραλλαγή. Επίσης σημαντικό είναι ότι η sqrt_class είναι περίπου στα ίδια ποσοστά με την const. Επίσης αξιοσημείωτο είναι ότι η μέθοδος cbrt είναι πολύ καλύτερη στα σύνολα με θόρυβο και έχει ίδιο ποσοστό σε αυτά τα dataset που δεν έχουν θόρυβο. Αυτό το οποίο βλέπουμε επίσης είναι πως η παραλλαγή Multipass τα καταφέρνει πολύ καλύτερα στα dataset που έχουν μεγαλύτερα ποσοστά θορύβου και από την cbrt αλλά και από την const. Σημαντικό να αναφερθεί είναι πως η μέθοδος AdaNN συνήθως είναι πιο χαμηλά από τον Best KNN. Επίσης σημαντικό είναι πως υπάρχουν πάρα πολλές εκδοχές οι οποίες πλησιάζουν πάρα πολύ στην κλασσική έκδοση του AdaNN αλλά και του Best-KNN. Αυτό το οποίο είναι σημαντικό όμως να αναφερθεί είναι πως τα κ τα οποία υπήρχαν για την τιμή του υπολογισμού του Best KNN έχουν έρθει από το άρθρο και μετά από διαδικασία cross validation και tuning. Χάρη στην διαδικασία όμως πολλών παραλλαγών γλιτώνουμε αυτήν την διαδικασία του tuning.

Ένα επίσης σημαντικο χαρακτηριστικό το οποίο είναι άξιο παρατήρησης είναι ο πίνακας 5.3 όπου βλέπουμε τα στατιστικά αφαίρεσης των προτύπων από κάθε σύνολο δεδομένων. Συγκεκριμένα στον πίνακα αυτόν βλέπουμε στην πρώτη στήλη το σύνολο των δεδομένων και στην δεύτερη στήλη τον αριθμό των στιγμιοτύπων που περιλαμβάνει. Στην πορεία αυτό που παρουσιάζεται είναι για την παραλλαγή AdaNN NR OnePass και την παραλλαγή AdaNN NR Multipass ότι ο αριθμός των στιγμιοτύπων που αφαιρέθηκαν από κάθε dataset καθώς και το τελικό ποσοστό αφαίρεσης των στιγμιοτύπων επί του συνολικού dataset. Συγκεκριμένα παρατηρούμε πως ο αλγόριθμος AdaNN NR Multipass αφαιρεί περισσότερα στιγμιότυπα από τον

AdaNN NR Onepass καθώς κάνει πολλαπλά περάσματα για αφαίρεση θορύβου. Επίσης άξιο παρατήρησης είναι ότι τα ποσοστά αφαίρεσης θορύβου σε κάθε dataset είναι αρκετά κοντά στα πραγματικά ποσοστά θορύβου που έχουν, όπως φαίνονται στα ονόματα τους.

Set	KNN-1	KNN-10	Best-KNN	const	sqrt	cbrt	sqrt_class	NR OnePass	NR MultiPass
bl	0.784	0.872	0.888	0.864	0.864	0.864	0.856	0.864	0.84
bl10	0.72	0.872	0.888	0.864	0.88	0.856	’0.864	0.84	0.832
bl30	0.584	0.784	0.864	0.728	0.752	0.744	’0.752	0.736	0.776
bn	0.872	0.901	0.906	0.897	0.897	0.897	0.897	0.897	0.897
ecl	0.835	0.910	0.910	0.925	0.895	0.925	0.910	0.895	0.880
ecl10	0.776	0.895	0.910	0.895	0.880	0.895	0.895	0.910	0.880
ecl30	0.611	0.865	0.835	0.865	0.835	0.865	0.880	0.850	0.850
iris	0.9	0.933	0.933	0.933	0.933	0.933	0.933	0.933	0.933
lir	0.957	0.940	0.947	0.954	0.952	0.953	0.954	0.952	0.951
ls	0.899	0.905	0.909	0.907	0.904	0.904	0.904	0.906	0.903
ls10	0.816	0.906	0.906	0.903	0.895	0.900	0.898	0.903	0.903
ls30	0.612	0.883	0.886	0.826	0.836	0.840	0.840	0.837	0.859
mgt	0.801	0.835	0.835	0.827	0.822	0.823	0.823	0.823	0.822
mgt10	0.732	0.829	0.830	0.800	0.801	0.803	0.801	0.802	0.807
pd	0.990	0.986	0.990	0.988	0.987	0.988	0.988	0.989	0.989
pd10	0.893	0.987	0.988	0.986	0.980	0.986	0.984	0.988	0.989
pd30	0.697	0.983	0.987	0.956	0.953	0.962	0.960	0.972	0.977
ph	0.887	0.839	0.887	0.885	0.889	0.889	0.889	0.887	0.884
ph10	0.813	0.846	0.848	0.862	0.854	0.857	0.855	0.858	0.861
ph30	0.652	0.772	0.799	0.712	0.725	0.725	0.726	0.733	0.745
pm	0.771	0.790	0.745	0.771	0.784	0.764	0.784	0.784	0.777
pm10	0.725	0.784	0.738	0.751	0.732	0.745	0.758	0.745	0.745
pm30	0.640	0.686	0.738	0.666	0.640	0.653	0.647	0.653	0.692
tn	0.955	0.978	0.979	0.963	0.964	0.963	0.964	0.962	0.963
tn10	0.843	0.970	0.977	0.941	0.943	0.942	0.943	0.943	0.95
tn30	0.672	0.872	0.975	0.792	0.821	0.814	0.820	0.802	0.834
txr	0.987	0.976	0.987	0.986	0.986	0.986	0.986	0.987	0.987
txr10	0.905	0.976	0.974	0.982	0.979	0.982	0.981	0.982	0.985
txr30	0.697	0.972	0.972	0.95	0.944	0.95	0.949	0.966	0.975
ys	0.493	0.574	0.587	0.564	0.560	0.557	0.577	0.543	0.560
ys10	0.449	0.577	0.581	0.540	0.540	0.550	0.533	0.527	0.557
ys30	0.368	0.554	0.574	0.510	0.503	0.516	0.527	0.513	0.510

Πίνακας 5.2: Μετρική Ακρίβειας για κάθε παραλλαγή

Set	Size	Method 1	Noise Removed	%	Method 2	Noise Removed	%
bl	500	NR OnePass	42	8.40	Ada NR Multipass	44	8.80
bl10	500	NR OnePass	69	13.80	Ada NR Multipass	82	16.40
bl30	500	NR OnePass	120	24.00	Ada NR Multipass	160	32.00
bn	4240	NR OnePass	196	4.62	Ada NR Multipass	232	5.47
ecl	269	NR OnePass	37	13.75	Ada NR Multipass	43	15.99
ecl10	269	NR OnePass	58	21.56	Ada NR Multipass	65	24.16
ecl30	269	NR OnePass	105	39.03	Ada NR Multipass	118	43.87
iris	120	NR OnePass	5	4.17	Ada NR Multipass	5	4.17
lir	16000	NR OnePass	190	1.19	Ada NR Multipass	213	1.33
ls	5148	NR OnePass	202	3.92	Ada NR Multipass	250	4.86
ls10	5148	NR OnePass	644	12.51	Ada NR Multipass	709	13.77
ls30	5148	NR OnePass	1477	28.69	Ada NR Multipass	1620	31.47
mgt	15216	NR OnePass	1156	7.60	Ada NR Multipass	1442	9.48
mgt10	15216	NR OnePass	1799	11.82	Ada NR Multipass	2251	14.79
pd	8794	NR OnePass	29	0.33	Ada NR Multipass	29	0.33
pd10	8794	NR OnePass	918	10.44	Ada NR Multipass	926	10.53
pd30	8794	NR OnePass	2514	28.59	Ada NR Multipass	2586	29.41
ph	4324	NR OnePass	140	3.24	Ada NR Multipass	177	4.09
ph10	4324	NR OnePass	364	8.42	Ada NR Multipass	430	9.94
ph30	4324	NR OnePass	546	12.63	Ada NR Multipass	743	17.18
pm	615	NR OnePass	68	11.06	Ada NR Multipass	94	15.28
pm10	615	NR OnePass	82	13.33	Ada NR Multipass	116	18.86
pm30	615	NR OnePass	101	16.42	Ada NR Multipass	143	23.25
tn	5920	NR OnePass	61	1.03	Ada NR Multipass	65	1.10
tn10	5920	NR OnePass	502	8.48	Ada NR Multipass	579	9.78
tn30	5920	NR OnePass	806	13.61	Ada NR Multipass	1187	20.05
txr	4400	NR OnePass	23	0.52	Ada NR Multipass	24	0.55
txr10	4400	NR OnePass	443	10.07	Ada NR Multipass	447	10.16
txr30	4400	NR OnePass	1269	28.84	Ada NR Multipass	1296	29.45
ys	1188	NR OnePass	275	23.15	Ada NR Multipass	314	26.43
ys10	1188	NR OnePass	353	29.71	Ada NR Multipass	386	32.49
ys30	1188	NR OnePass	528	44.44	Ada NR Multipass	570	47.98

Πίνακας 5.3: Περιγραφή Αποτελεσμάτων Παραλλαγών Αφαίρεσης Θορύβου

Κεφάλαιο 6

Συμπεράσματα και Μελλοντική έρευνα

6.1 Συμπεράσματα

Είναι φανερό πως ο κλασσικός αλγόριθμος του KNN χρειαζόταν μια διαφορετική προσέγγιση όσον αφορά την σταθερότητα της τιμής του κ. Γενικά όπως αναλύθηκε υπάρχει μια πληθώρα από παραλλαγές του KNN αλλά ήταν φανερό πως η κλασσική του εκδοχή θα έπρεπε σίγουρα να διαφοροποιηθεί όσον αφορά την δυναμικότητα της τιμής. Τηρώντας πάντα την ίδια μετρική απόστασης με τον παραδοσιακό αλγόριθμο KNN, δηλαδή την ευκλείδια απόσταση στην παρούσα διπλωματική υλοποιήθηκε η παραλλαγή του AdaNN η οποία ορίζει μια δυναμικότητα στον προσδιορισμό του κ για κάθε στιγμιότυπο. Το πρόβλημα με αυτήν την επίσης κλασσική εκδοχή του AdaNN όπως παρουσιαζόταν στο άρθρο [5] είναι ότι η μεταβλητή τιμή του κάθε στιγμιότυπου έφτανε μέχρι σε ένα συγκεκριμένο όριο. Στην παρούσα διπλωματική αναλύθηκαν διάφορες παραλλαγές με τις οποίες μπορεί να υπολογιστεί αυτό το μέγιστο όριο σύμφωνα με διάφορους παράγοντες όπως ο αριθμός των στιγμιοτύπων του dataset καθώς και ο αριθμος των στιγμοτύπων της κλάσης στην οποία ανήκει το στιγμιότυπο. Ένα άλλο πρόβλημα το οποίο λύθηκε είναι η διαχείριση των τιμών αυτών των προβληματικών στιγμιοτύπων τα οποία ξεπερνούσαν αυτό το λογικό μέγιστο αριθμό των γειτόων ώστε να κατηγοριοποιηθεί σωστά. Στις απλές παραλλαγές απλά έγινε ορισμός απλά του κ σαν την τιμή του μέγιστου ορίου. Στις παραλλαγές διαχείρισης θορύβου θεωρούμε πως υπάρχουν στιγμιότυπα τα οποία εφόσον δεν μπορούν να κατηγοριοποιηθούν με έναν λογικό αριθμο κ θα τα θεωρήσουμε ότι είναι θόρυβος. Για αυτόν τον λόγο υλοποιήθηκαν και 2 παραλλαγές του αλγορίθμου η NR OnePass και NR Multipass η οποίες θα κάνουν ένα πέρασμα αφαίρεσης θορύβου αλλά και πολλαπλά περάσματα για να αφαιρεθούν αυτά τα στιγμιότυπα. Αυτό το οποίο φάνηκε μετά από την υλοποίηση των παραλλαγών αυτών στην γλώσσα Python είναι πως υπάρχουν πολλές παραλλαγές οι οποίες θα πλησίασουν πρώτα την κλασσική εκδοχή του αλγορίθμου AdaNN αλλά πολλές φορές και την τιμή του Best KNN. Αυτό είναι ιδιαίτερα σημαντικό καθώς αυτή η εγγύτητα στην τιμή του Best KNN προκύπτει από τον κώδικα της παραλλαγής χωρίς να κάνουμε την διαδικασία του tuning της παραμέτρου κ. Επίσης σημαντικό είναι ότι πολλές παραλλαγές πετυχαίνουν πάρα πολύ καλά αποτελέσματα στην αφαίρεση του θορύβου στα dataset τα οποία έχουν μεγαλύτερα ποσοστά θορύβου. Όπως φάινονται και στα αποτελέσματα το ποσοστό αφαίρεσης του θορύβου σε κάθε dataset είναι πάρα πολύ κοντά στο αναμενόμενο θόρυβο που έχει το dataset.

6.2 Μελλοντική έρευνα

Κατά την διάρκεια της διπλωματικής έγινε ιδιαίτερη αναφορά και στον αλγόριθμο KNN και στον αλγόριθμο AdaNN καθώς και έγινε ιδιαίτερη διερεύνηση στις διάφορες παραλλαγές τους. Για αυτόν τον λόγο αναφέρθηκαν διάφορες ιδιότητες που έχει και ο AdaNN αλλά και οι υλοποιημένες παραλλαγές. Ιδιαίτερο ενδιαφέρον για μελλοντική έρευνα παρουσιάζουν οι παραλλαγές οι οποίες έχουν να κάνουν με την αφαίρεση του θορύβου. Για αυτόν τον λόγο μια πιθανή μελλοντική έρευνα είναι να γίνει διερεύνηση των ποσοστών ακρίβειας πάνω στην αφαίρεση του θορύβου σε σύγκριση αυτών των παραλλαγών με κάποιους άλλους παραδοσιακούς αλγορίθμους που έχουν ως στόχο την αφαίρεση του θορύβου. Επίσης σημαντικό επίσης θα είναι να γίνει διερεύνηση του χρόνου εκτέλεσης του AdaNN καθώς και των νέων παραλλαγών.

Βιβλιογραφία

[1] I. Vlahavas, P. Kefalas, N. Bassiliades, F. Kokkoras, and S. Ilias, *Τεχνητή Νοημοσύνη - Γ' Έκδοση*. University of Macedonia Press, 2011.

[2] E. Pitoura, "Εισαγωγή [Σημειώσεις Διάλεξης]. Εξόρυξη Δεδομένων.." https://www.cs.uoi.gr/~pitoura/courses/dm/introspring11.pdf, 2011.

[3] S. Ougiaroglou, "Προ-επεξεργασία δεδομένων & Κατηγοριοποίηση [Σημειώσεις Διάλεξης]. Εξόρυξη Δεδομένων.." https://moodle.teithe.gr/mod/folder/view.php?id=36603, 2020.

[4] Canhasi, Written by: Ercan, "Lazy vs. eager learning." https://www.baeldung.com/cs/lazy-vs-eager-learning, 2023. Accessed on September 1, 2023.

[5] 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, Aug 2010.

[6] R. J. Roiger and M. W. Geatz, *Εξόρυξη πληροφορίας, Ένας εισαγωγικός οδηγός με παραδείγματα*. Κλειδάριθμος, 2008.

[7] Kilian Q. Weinberger, "Lecture 2: K-nearest neighbors." https://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote02_kNN.html, 2023. Accessed on September 1, 2023.

[8] "K-nearest neighbors at ibm." https://www.ibm.com/topics/knn, 12-09-2023.

[9] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation*, pp. 532–538. Boston, MA: Springer US, 2009.

[10] M. M. Deza and E. Deza, *Encyclopedia of distances*. Springer, 2009.

[11] "Manhattan distance." https://www.sciencedirect.com/topics/mathematics/manhattan-distance, 2023. Accessed on September 1, 2023.

[12] J. Sun, W. Du, and N. Shi, "A survey of knn algorithm," *Information Engineering and Applied Computing*, vol. 1, 05 2018.

[13] S. Dhanabal and C. SA, "A review of various k-nearest neighbor query processing techniques," *Int. J. Comput. Appl.*, vol. 3, 01 2011.

[14] E. Fox, "Nn search with kd-trees." https://www.coursera.org/lecture/ml-clustering-and-retrieval/nn-search-with-kd-trees-6eTzw, 12-09-2023.

[15] P. Kraus and W. Dzwinel, "Nearest neighbor search by using partial kd-tree method," *TACS*, vol. 20, pp. 149–165, 01 2008.

[16] C. Ransome and C. Ransome, *The Fermi Paradox and Galactic Habitability (Masters thesis)*. PhD thesis, 04 2017.

[17] J. Sun, W. Du, and N. Shi, "A survey of knn algorithm," *Information Engineering and Applied Computing*, vol. 1, 05 2018.

[18] Z. Voulgaris and G. Magoulas, "Extensions of the k nearest neighbour methods for classification problems," *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, AIA 2008*, 02 2008.

[19] Python Software Foundation, "Python documentation." https://docs.python.org/3/license.html, 2023. Accessed on September 1, 2023.

[20] https://numpy.org/, 2023. Accessed on September 1, 2023.

[21] https://towardsdatascience.com/how-fast-numpy-really-is-e9111df44347, 2023. Accessed on September 1, 2023.

[22] https://sci2s.ugr.es/keel/, 2023. Accessed on September 20, 2023.

[23] S. Ougiaroglou, G. Evangelidis, and K. Diamantaras, *Dynamic k-NN Classification Based on Region Homogeneity*, pp. 27–37. 08 2020.

[24] G. S. K. Ranjan, A. Verma, and R. Sudha, "K-nearest neighbors and grid search cv based real time fault monitoring system for industries," pp. 1–5, 03 2019.

Παράρτημα Α΄

Παράρτημα: Παραλλαγές Χωρίς Αφαίρεση Θορύβου

Α΄.1 AdaNN

```
from sklearn.neighbors import KNeighborsClassifier
from pandas import read_csv
from sklearn import metrics
import numpy as np
import math
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from IPython import get_ipython

import sys

#Experiment Inputs


class AdaNN:


    def __init__(self, k_limit_fuction = 'const'):
        self.k_limit_fuction = k_limit_fuction
        print('\t k limit function set = ' + k_limit_fuction)



    def calculateMaximumKLimit(self, y_test, y_train, numberOfInstances):

        if self.k_limit_fuction == 'sqrt' :
            return  round(math.sqrt(numberOfInstances))
        elif self.k_limit_fuction == 'cbrt' :
            return round(numberOfInstances ** (1/3))
        elif self.k_limit_fuction == 'sqrt_limit_class' :

            testingInstanceClass =  y_test[0];
```

```
            return round(np.sqrt(y_train.count(testingInstanceClass)) + 1)
        else:
            return  9


    def calculateBestK(self, x_train, y_train, x_test, y_test):
        y_test = [y_test]
        x_test = x_test.reshape(1,-1)

        numberOfTrainingInstances = len(x_train)

        k_limit = self.calculateMaximumKLimit(y_test, y_train,
                                              numberOfTrainingInstances)

        print('\t \t k limit = ' + str(k_limit))

        k = 1
        while True:

            knn = KNeighborsClassifier(n_neighbors=k)
            knn.fit(x_train, y_train)
            y_pred = knn.predict(x_test)

            if metrics.accuracy_score(y_test, y_pred) == 1.0 :
                 break

            if k == self.calculateMaximumKLimit(y_test, y_train,
                                                numberOfTrainingInstances):
                return self.calculateMaximumKLimit(y_test, y_train,
                                                   numberOfTrainingInstances
                                                   )

                break
            k+=1
        return k

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


    def method(self, folder, trainingDataSetFileName, dSep):

        data = read_csv(folder + trainingDataSetFileName, header=None, sep =
                                              dSep).values


        x=data[:, 0:data.shape[1]-1].astype(float) #get all but last column, (
                                              attributes)
        y=data[:, data.shape[1]-1] #get last column,  targets

        datasetWithK = open(folder + str("k_") + trainingDataSetFileName, "a")


        i = 0

        while i < x.shape[0]: #number of instances
```

```
            x_train = list(x)[:i] + list(x)[i+1:]
            x_test = x[i]
            y_train = list(y)[:i] + list(y)[i+1:]
            y_test = y[i]

            k = self.calculateBestK(x_train, y_train, x_test, y_test);

            newLine = ""
            for j in range(x.shape[1]):
                newLine += str(x_test[j]) + dSep
            newLine += str(int(y[i])) + dSep + str(k) + "\n"
            datasetWithK.write(newLine)

            print(i)

            i=i+1



        datasetWithK.close()

        print('\t Done' + folder + '\n')
        get_ipython().magic('reset -sf')


#=========================================================
```

Α΄.2 ApplyAdaNN

```
from sklearn.neighbors import KNeighborsClassifier
from pandas import read_csv
from sklearn import metrics
from sklearn.neighbors import NearestNeighbors
import numpy as np
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from IPython import get_ipython


# define a function to calculate custom weights
def my_weight_func(distances):
    # create an array of ones with the same shape as distances
    weights = [1] * len(distances)
    # find the index of the closest neighbor
    closest_idx = distances.argmin()
    # assign a weight of 1.1 to the closest neighbor
    weights[closest_idx] = 1.1
    return weights


def find_indexOf_nearest_instance_of_model(xTestInstance, xModel):
```

```
    neigh = NearestNeighbors(n_neighbors=1)
    neigh.fit(xModel)
    index = neigh.kneighbors(xTestInstance, return_distance=False)
    return index[0][0]

def method1(folder, testDatasetName, modelFileName, dSep):
    #every line in modelFile has this format x1 x2  x3  x4  y k
    #modelFileName = "k_data1"
    #every line in testFile has this format x1  x2  x3  x4  y
    #testDatasetName = "data_te1"

    #dSep = ' '



    modelData = read_csv(folder + modelFileName, header=None, sep = dSep).
                                            values
    testData = read_csv(folder + testDatasetName, header=None, sep = dSep).
                                            values

    xModel = modelData[:, 0:modelData.shape[1] - 2].astype(float)
    yModel = modelData[:, modelData.shape[1] - 2].astype(int)
    k = modelData[:, modelData.shape[1] - 1].astype(int)

    xTestData = testData[:, 0:testData.shape[1] -1].astype(float)
    yTestData = testData[:, testData.shape[1] -1].astype(int)

    predictionsForTestData = list()

    for xTestInstance in xTestData:
        xTestInstance = xTestInstance.reshape(1,-1) #transform array from 2D
                                            to 1D

        nearestNeigborIndex = find_indexOf_nearest_instance_of_model(
                                            xTestInstance, xModel)

        #We should not allow k greater than the number of instances of model
        kForTestInstance = k[nearestNeigborIndex] if k[nearestNeigborIndex] <
                                            k.shape[0] else k.shape[0]

        knn = KNeighborsClassifier(n_neighbors=kForTestInstance, weights =
                                            my_weight_func)
        knn.fit(xModel, yModel)

        y_pred = knn.predict(xTestInstance)
        predictionsForTestData.append(y_pred)

    precision, recall, fscore, supp = metrics.precision_recall_fscore_support(
                                            yTestData, predictionsForTestData,
                                            average=None)

    scores_tuple = (np.mean(precision),
                    np.mean(recall),
                    np.mean(fscore),
```

```
                    np.mean(supp),
                    metrics.accuracy_score(yTestData, predictionsForTestData))

    return scores_tuple
```

Α΄.3 Testing

```
from AdaNN import *
from ApplyAdaNN import *
from KNN import predict_and_evaluate
import os
import xlsxwriter
import sys

#Set a logger
class Logger(object):
    def __init__(self, filename="Default.log"):
        self.terminal = sys.stdout
        self.log = open(filename, "a")

    def write(self, message):
        self.terminal.write(message)
        self.log.write(message)
        self.log.flush()

    def flush(self):
        pass

sys.stdout = Logger("MYLOGS.txt")


if __name__ == '__main__':


    path_to_datasets = './Datasets/data'

    workbook = xlsxwriter.Workbook('AdaNN.xlsx')
    worksheet = workbook.add_worksheet()

    dataset_folders = os.listdir(path_to_datasets)

    #Write headers of file

    worksheet.write(0, 0, 'Dataset')
    worksheet.write(0, 1, 'Limit Function')

    worksheet.write(0, 3, 'Avg Recall')
    worksheet.write(0, 4, 'Avg Precision')
    worksheet.write(0, 5, 'Avg Fscore')
```

```
    worksheet.write(0, 6, 'Avg Support')
    worksheet.write(0, 7, 'Accuracy')


    k_array = {
    "bl": 42,
    "bl10": 43,
    "bl30": 32,
    "bn": 29,
    "ecl": 6,
    "ecl10": 6,
    "ecl30": 14,
    "iris": 11,
    "lir": 4,
    "ls": 8,
    "ls10": 8,
    "ls30": 13,
    "mgt": 10,
    "mgt10": 20,
    "pd": 1,
    "pd10": 8,
    "pd30": 12,
    "ph": 1,
    "ph10": 8,
    "ph30": 48,
    "pm": 48,
    "pm10": 18,
    "pm30": 47,
    "sh": 1,
    "tn": 49,
    "tn10": 47,
    "tn30": 50,
    "txr": 1,
    "txr10": 5,
    "txr30": 10,
    "ys": 14,
    "ys10": 13,
    "ys30": 14,
}

#     k_array = {
#     "bl": 1,
#     "bl10": 1,
#     "bl30":  1,
#     "bn":  1,
#     "ecl": 1,
#     "ecl10":  1,
#     "ecl30":  1,
#     "iris":  1,
#     "lir":  1,
#     "ls":  1,
#     "ls10":  1,
#     "ls30":  1,
#     "mgt":  1,
```

```
#     "mgt10":  1,
#     "pd": 1,
#     "pd10":  1,
#     "pd30":  1,
#     "ph":  1,
#     "ph10":  1,
#     "ph30":  1,
#     "pm":  1,
#     "pm10":  1,
#     "pm30":  1,
#     "sh": 1,
#     "tn":  1,
#     "tn10":  1,
#     "tn30":  1,
#     "txr": 1,
#     "txr10":  1,
#     "txr30":  1,
#     "ys":  1,
#     "ys10":  1,
#     "ys30":  1,
# }

#     k_array = {
#     "bl": 10,
#     "bl10": 10,
#     "bl30":  10,
#     "bn":  10,
#     "ecl": 10,
#     "ecl10":  10,
#     "ecl30":  10,
#     "iris":  10,
#     "lir":  10,
#     "ls":  10,
#     "ls10":  10,
#     "ls30": 10,
#     "mgt":  10,
#     "mgt10":  10,
#     "pd": 10,
#     "pd10":  10,
#     "pd30":  10,
#     "ph":  10,
#     "ph10":  10,
#     "ph30":  10,
#     "pm":  10,
#     "pm10":  10,
#     "pm30":  10,
#     "sh": 10,
#     "tn":  10,
#     "tn10": 10,
#     "tn30":  10,
#     "txr": 10,
#     "txr10":  10,
#     "txr30":  10,
#     "ys":  10,
```

```
#     "ys10":  10,
#     "ys30": 10,
# }
    available_functions = ['sqrt', 'cbrt', 'const', 'sqrt_limit_class']
    excelRowIndex = 0


    for folder in dataset_folders:


        for func in available_functions:

            excelRowIndex = excelRowIndex + 2 #

            print('Building model')


            model = AdaNN(func)

            print('Working on dataset...' + folder)

            model.method(path_to_datasets  + '/' + folder + '/',  'data1', '
                                                    ')

            print('Applying on dataset...' + folder)

            scores_tuple  = method1(path_to_datasets + '/' + folder + '/',  '
                                                   data_te1', 'k_data1',' ')

            precision, recall, fscore, supp, accuracy = scores_tuple


            print("---------------Algorithm Results--------------------------
                                                    ")
            print(scores_tuple)

            #---------------Algorithm Results--------------------------

            print('Writing results to excel file...')


            worksheet.write(excelRowIndex, 0, str(folder))
            worksheet.write(excelRowIndex, 1, str(func))


            worksheet.write(excelRowIndex, 3, str(recall))
            worksheet.write(excelRowIndex, 4, str(precision))
            worksheet.write(excelRowIndex, 5, str(fscore))
            worksheet.write(excelRowIndex, 6, str(supp))
            worksheet.write(excelRowIndex, 7, str(accuracy))


            os.remove(path_to_datasets  + '/' + folder + '/k_data1')
```

```
        # print("-------------------------------------------------------------")


        print("---------------KNN-----------------------------------------")


        training = path_to_datasets  + '/' + folder + '/data1'
        testing = path_to_datasets + '/' + folder + '/data_te1'

        scores_tuple  = predict_and_evaluate(training, testing, k_array[folder
                                             ])

        print(scores_tuple)

        precision, recall, fscore, supp, accuracy = scores_tuple

        excelRowIndex = excelRowIndex + 2 #

        worksheet.write(excelRowIndex, 0, str(folder))
        worksheet.write(excelRowIndex, 1, str('KNN'))


        worksheet.write(excelRowIndex, 3, str(recall))
        worksheet.write(excelRowIndex, 4, str(precision))
        worksheet.write(excelRowIndex, 5, str(fscore))
        worksheet.write(excelRowIndex, 6, str(supp))
        worksheet.write(excelRowIndex, 7, str(accuracy))

        print("-------------------------------------------------------------")

    workbook.close()
```

Παράρτημα Β′

Παράρτημα: Παραλλαγή AdaNN NR OnePass

Β′.1 AdaNN

```
from sklearn.neighbors import KNeighborsClassifier
from pandas import read_csv
from sklearn import metrics
import numpy as np
import math
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from IPython import get_ipython

import sys

#Experiment Inputs


class AdaNN:


    def __init__(self, k_limit_fuction = 'const'):
        self.k_limit_fuction = k_limit_fuction
        #print('\t k limit function set = ' + k_limit_fuction)



    def calculateMaximumKLimit(self, y_test, y_train):

        testingInstanceClass =  y_test[0]
        return round(np.sqrt(y_train.count(testingInstanceClass) + 1))



    def calculateBestK(self, x_train, y_train, x_test, y_test):
        y_test = [y_test]
```

```
        x_test = x_test.reshape(1,-1)

        numberOfTrainingInstances = len(x_train)

        k_limit = self.calculateMaximumKLimit(y_test, y_train)

        #print('\t \t k limit = ' + str(k_limit))

        k = 1
        while True:

            knn = KNeighborsClassifier(n_neighbors=k)
            knn.fit(x_train, y_train)
            y_pred = knn.predict(x_test)

            if metrics.accuracy_score(y_test, y_pred) == 1.0 :
                 break

            if k == self.calculateMaximumKLimit(y_test, y_train):
                return -1
                break
            k+=1
        return k

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


    def method(self, folder, trainingDataSetFileName, dSep):

        data = read_csv(folder + trainingDataSetFileName, header = None, sep =
                                                    dSep).values

        x = data[:, 0 : data.shape[1] - 1].astype(float) #get all but last
                                                    column, (attributes)
        y = data[:, data.shape[1] - 1] #get last column,  targets

        datasetWithK = open(folder + str("k_") + trainingDataSetFileName, "a")

        totalInstances = x.shape[0]
        instancesMarkedAsNoise = 0


        i = 0

        k_list = list() # list that stores k for all instances of training

        #for every instance we calculate each k and add it to the list
        while i < x.shape[0]: #number of instances
           x_train = list(x)[:i] + list(x)[i+1:]
           x_test = x[i]
           y_train = list(y)[:i] + list(y)[i+1:]
           y_test = y[i]

           k = self.calculateBestK(x_train, y_train, x_test, y_test)
```

```
            k_list.append(k)
            i = i + 1

        #convert list to array
        k_array = np.array(k_list)

        #add the collumn of the k to the data
        data_with_k = np.column_stack((data, k_array))


        #for every instance we check if k > 0 (not marked as noise)
        #and we write the line to the new k_trainDatasetFileName file
        #line format: x1 x2 x3 x4 y k
        i = 0
        while i < data_with_k.shape[0]:

            x_train = list(x)[:i] + list(x)[i+1:]
            x_test = x[i]
            y_train = list(y)[:i] + list(y)[i+1:]
            y_test = y[i]

            if (k_array[i] > 0):

                newLine = ""
                for j in range(x.shape[1]):
                    newLine += str(x_test[j]) + dSep
                newLine += str(int(y[i])) + dSep + str(k_array[i]) + "\n"
                datasetWithK.write(newLine)
            else:
                instancesMarkedAsNoise = instancesMarkedAsNoise + 1

            i=i+1


        datasetWithK.close()
        print('\t Done' + folder + '\n')
        get_ipython().magic('reset -sf')

        noise_removal_details = (instancesMarkedAsNoise, totalInstances)
        return noise_removal_details


#=======================================================
```

Β΄.2 Testing

```
from AdaNN import *
from ApplyAdaNN import *
import os
```

```
import xlsxwriter
import sys

#Set a logger
class Logger(object):
    def __init__(self, filename="Default.log"):
        self.terminal = sys.stdout
        self.log = open(filename, "a")

    def write(self, message):
        self.terminal.write(message)
        self.log.write(message)
        self.log.flush()

    def flush(self):
        pass

sys.stdout = Logger("MYLOGS.txt")


if __name__ == '__main__':


    path_to_datasets = './Datasets/data'

    workbook = xlsxwriter.Workbook('AdaNN.xlsx')
    worksheet = workbook.add_worksheet()

    dataset_folders = os.listdir(path_to_datasets)

    #Write headers of file

    worksheet.write(0, 0, 'Dataset')
    worksheet.write(0, 1, 'Limit Function')

    worksheet.write(0, 3, 'Avg Recall')
    worksheet.write(0, 4, 'Avg Precision')
    worksheet.write(0, 5, 'Avg Fscore')
    worksheet.write(0, 6, 'Avg Support')
    worksheet.write(0, 7, 'Accuracy')

    worksheet.write(0, 8, 'Noise')
    worksheet.write(0, 9, 'Dataset Size')
    worksheet.write(0, 10, 'Noise Percentage')


    available_functions = ['sqrt_limit_class']
    excelRowIndex = 0


    for folder in dataset_folders:

        for func in available_functions:
```

```
        excelRowIndex = excelRowIndex + 2 #

        print('Building model')


        model = AdaNN(func)

        print('Working on dataset...' + folder)

        noise_removal_tuple = model.method(path_to_datasets  + '/' +
                                           folder + '/',  'data1', '
                                           ')

        print('Applying on dataset...' + folder)

        scores_tuple  =  method1(path_to_datasets + '/' + folder + '/',  '
                                           data_te1', 'k_data1','  ')

        precision, recall, fscore, supp, accuracy = scores_tuple
        instancesMarkedAsNoise, totalInstances = noise_removal_tuple


        print("---------------Algorithm Results---------------------------
                                           ")
        print(scores_tuple)

        #---------------Algorithm Results---------------------------

        print('Writing results to excel file...')


        worksheet.write(excelRowIndex, 0, str(folder))
        worksheet.write(excelRowIndex, 1, str(func))


        worksheet.write(excelRowIndex, 3, str(recall))
        worksheet.write(excelRowIndex, 4, str(precision))
        worksheet.write(excelRowIndex, 5, str(fscore))
        worksheet.write(excelRowIndex, 6, str(supp))
        worksheet.write(excelRowIndex, 7, str(accuracy))

        worksheet.write(excelRowIndex, 8, str(instancesMarkedAsNoise))
        worksheet.write(excelRowIndex, 9, str(totalInstances))
        worksheet.write(excelRowIndex, 10, str(instancesMarkedAsNoise /
                                           totalInstances * 100))


        #os.remove(path_to_datasets  + '/' + folder + '/k_data1')

    print("------------------------------------------------------------")



workbook.close()
```

Παράρτημα Γ′

Παράρτημα: Παραλλαγή AdaNN NR Multipass

Γ′.1 AdaNN

```
from sklearn.neighbors import KNeighborsClassifier
from pandas import read_csv
from sklearn import metrics
import numpy as np
import math
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from IPython import get_ipython

import sys

#Experiment Inputs


class AdaNN:


    def __init__(self, k_limit_fuction = 'const'):
        self.k_limit_fuction = k_limit_fuction
        #print('\t k limit function set = ' + k_limit_fuction)



    def calculateMaximumKLimit(self, y_test, y_train):

        testingInstanceClass =  y_test[0]
        return round(np.sqrt(y_train.count(testingInstanceClass) + 1))



    def calculateBestK(self, x_train, y_train, x_test, y_test):
        y_test = [y_test]
```

```
        x_test = x_test.reshape(1,-1)

        numberOfTrainingInstances = len(x_train)

        k_limit = self.calculateMaximumKLimit(y_test, y_train)

        #print('\t \t k limit = ' + str(k_limit))

        k = 1
        while True:

            knn = KNeighborsClassifier(n_neighbors=k)
            knn.fit(x_train, y_train)
            y_pred = knn.predict(x_test)

            if metrics.accuracy_score(y_test, y_pred) == 1.0 :
                 break

            if k == self.calculateMaximumKLimit(y_test, y_train):
                return -1
                break
            k+=1
        return k

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


    def method(self, folder, trainingDataSetFileName, dSep):

        data = read_csv(folder + trainingDataSetFileName, header=None, sep =
                                                dSep).values


        x=data[:, 0:data.shape[1]-1].astype(float) #get all but last column, (
                                                attributes)
        y=data[:, data.shape[1]-1] #get last column,  targets

        datasetWithK = open(folder + str("k_") + trainingDataSetFileName, "a")


        totalInstances = x.shape[0]
        instancesMarkedAsNoise = 0

        noiseExists = True
        k_array = []
        while noiseExists:

            i = 0
            k_list = list()
            # We calculate k for each instance of training data
            while i < x.shape[0]: #number of instances
                x_train = list(x)[:i] + list(x)[i+1:]
                x_test = x[i]
                y_train = list(y)[:i] + list(y)[i+1:]
```

```
            y_test = y[i]

            k = self.calculateBestK(x_train, y_train, x_test, y_test)
            k_list.append(k)
            i = i + 1


        i = 0
        noiseInstancesFound = 0
        #
        while i < x.shape[0]: #number of instances
            x_train = list(x)[:i] + list(x)[i+1:]
            x_test = x[i]
            y_train = list(y)[:i] + list(y)[i+1:]
            y_test = y[i]

            if (k_list[i] < 0):
                noiseInstancesFound = noiseInstancesFound + 1

                x = np.delete(x, i, 0) # delete the i element of 0 axis (
                                                        rows)
                y = np.delete(y, i, 0)  # delete the i element of 0 axis (
                                                        rows)
                k_list = np.delete(k_list, i, 0)  # delete the i element
                                                        of 0 axis (rows)
                instancesMarkedAsNoise = instancesMarkedAsNoise + 1
                i = i - 1

            i = i + 1

        if (noiseInstancesFound == 0):
            noiseExists = False
            k_array = k_list
        else:
            noiseExists = True



    # We check every k calculated
    # and if it is not marked as noise we write it to the file
    i = 0
    while i < x.shape[0]:

        x_train = list(x)[:i] + list(x)[i+1:]
        x_test = x[i]
        y_train = list(y)[:i] + list(y)[i+1:]
        y_test = y[i]

        if (k_array[i] > 0):

            newLine = ""
            for j in range(x.shape[1]):
                newLine += str(x_test[j]) + dSep
```

```
                newLine += str(int(y[i])) + dSep + str(k_array[i]) + "\n"
                datasetWithK.write(newLine)
            i=i+1


        datasetWithK.close()

        get_ipython().magic('reset -sf')

        print('\t Done' + folder + '\n')

        noise_removal_details = (instancesMarkedAsNoise, totalInstances)
        return noise_removal_details



#=========================================================
```