Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σύγκριση αλγορίθμων εξόρυξης γνώσης από πολύ μεγάλες βάσεις δεδομένων** **Comparison of data mining algorithms from very large databases**
Ονοματεπώνυμο Φοιτητή	**Τσουμπού Παναγιώτα**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**11002**
Επιβλέπων	**Γιάννης Θεοδωρίδης, Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2015**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Νίκος Πελέκης Επίκουρος Καθηγητής	Χαράλαμπος Κωνσταντόπουλος Επίκουρος Καθηγητής	Άγγελος Πικράκης Επίκουρος Καθηγητής

Πίνακας Περιεχομένων

ΠΕΡΙΛΗΨΗ

Ο αιώνας που διανύουμε είναι αδιαμφισβήτητα ο αιώνας της πληροφορίας. Η ανάπτυξη του Internet και η χρήση του στην καθημερινότητα μας, δημιούργησε ένα χώρο όπου τεράστιες ποσότητες πληροφορίας προστίθενται καθημερινά. Σύμφωνα με το ερευνητικό τμήμα της IBM εκτιμάται ότι μόνο στο μέσο κοινωνικής δικτύωσης Facebook, προστίθενται κάθε μέρα 100 Terabytes πληροφορίας. Επίσης εκτιμάται ότι το 2020 ο όγκος διακίνησης πληροφορίας στα μέσα κοινωνικής δικτύωσης θα ξεπερνά τα 35 Zettabytes. Για να κατανοήσουμε πόσο μεγάλος είναι αυτός ο όγκος πληροφορίας αξίζει να αναφέρουμε ότι 1 Zettabyte είναι ισο με 10^{21} bytes ή ίσο με 10^{12} Gigabytes. Αν και αυτός ο όγκος δεδομένων φαντάζει εξωπραγματικός, αξίζει να σημειωθεί ότι αποτελεί ένα πολύ μικρό ποσοστό των συνολικών δεδομένων που θα διακινούνται μέσω του διαδικτύου εφόσον η ιδέα του Internet of Things (IoT) έχει ήδη αρχίσει να γίνεται πραγματικότητα.

Δυστυχώς η ύπαρξη δεδομένων δεν συνεπάγεται και την ύπαρξη γνώσης "We are drowning in data, but starving for knowledge -- anonymous". Έτσι για να μετατραπούμε από κοινωνία πληροφορίας σε κοινωνία γνώσης χρειάζεται να βρούμε γρήγορους και αποδοτικούς τρόπους διαχείρισης και ανάλυσης οι οποίοι θα μπορούν να επεξεργαστούν με ταχύτητα και να εξάγουν αξιόπιστη γνώση από αυτούς τους τεράστιους όγκους δεδομένων. Σήμερα πολλές ερευνητικές ομάδες έχουν στραφεί προς αυτή την κατεύθυνση προσπαθώντας να συνεισφέρουν στον μετασχηματισμό του μεγάλου όγκου πληροφορίας σε γνώση. Ένας από τους πολλά υποσχόμενους τομείς για την εξαγωγή γνώσης από μεγάλους όγκους δεδομένων είναι ο τομέας της εξόρυξης δεδομένων.

Κατά την διάρκεια των τελευταίων ετών έχουν προταθεί πολλοί αλγόριθμοι οι οποίοι έχουν ως σκοπό την ανάλυση δεδομένων. Στις περισσότερες περιπτώσεις αυτοί οι αλγόριθμοι είναι πολύπλοκο να υλοποιηθούν και να εφαρμοστούν από έναν "απλό" χρήστη, γεγονός που κάνει την ανάλυση δεδομένων μία διαδικασία εξαιρετικά δύσκολη για μη ειδικούς. Για αυτό το λόγο έχουν αναπτυχθεί αρκετά πακέτα λογισμικού τα οποία είναι φιλικά ως προς τον χρήστη και του δίνουν την δυνατότητα να εφαρμόσει αυτούς τους αλγόριθμους στα δεδομένα του.

Σε αυτή τη διπλωματική εργασία παρουσιάζουμε τους αλγορίθμους ομαδοποίησης που εμπεριέχονται στο δημοφιλές λογισμικό ανάλυσης δεδομένων WEKA με σκοπό την μελέτη και την σύγκριση τους ως προς την δυνατότητά τους να διαχειρίζονται μεγάλα αρχεία δεδομένων. Επίσης υλοποιήσαμε και ενσωματώσαμε τον δημοφιλή αλγόριθμο CURE(Clustering Using REpresentatives) στο λογισμικό WEKA ο οποίος θεωρείται ένας από τους πιο πολλά υποσχόμενους αλγορίθμους εξόρυξης δεδομένων εφόσον παρέχει την δυνατότητα διαχείρισης μεγάλων όγκων δεδομένων και αναγνώρισης απομακρυσμένων σημείων (outliers). Μέσω μιας σειράς πειραμάτων παρουσιάζουμε για κάθε αλγόριθμο τα όρια επεξεργασίας δεδομένων με την χρήση του λογισμικού WEKA, καθώς και την ταχύτητα εκτέλεσης καθε ενός από αυτούς για διαφορετικές τιμές εγγραφών και χαρακτηριστικών.

ABSTRACT

The century we are living is undoubtedly the century of information. The growth of the Internet and its use in everyday life, created a space where huge amounts of information are daily added. The research department of IBM estimates that only social networking service Facebook, adds each day 100 Terabytes of information. It is also estimated that by 2020 the traffic volume of information in social media will exceed 35 Zettabytes. To understand how big this amount of information is, it is worth mentioning that 1 Zettabyte is equal to 10^{21} bytes or equal to 10^{12} Gigabytes. Although this amount of data seems unreal, it is worth noting that it is only a small percentage of the overall data that will be move via the internet because the idea of the Internet of Things (IOT) has already started to become reality.

Unfortunately, the fact that data exists does not means knowledge exists too "We are drowning in data, but starving for knowledge - anonymous". So in order to convert the society of information to society of knowledge, we need to find fast and efficient ways of management and analysis, which can fast extract reliable knowledge from huge volumes of data. Nowadays many research teams turn to this direction trying to contribute to the transformation of the large volume data into knowledge. One of the promising areas for extracting knowledge from large volumes of data is Data Mining.

The last few years many algorithms have been discovered in order to analyze data. In most cases these algorithms are complex to be implemented by a "simple" user, which makes data analysis an extremely difficult process for non-specialists. For this reason many user friendly software packages have been developed that allow the end user to apply these algorithms to his data.

In this thesis we present clustering algorithms which are included in the popular data analysis software WEKA in order to study and compare their ability to manage large data files. Also, we implemented and integrated the algorithm CURE (Clustering Using REpresentatives) into WEKA software, which is considered to be one of the most promising algorithms in data mining due to its ability to manage large volumes of data and identification of outliers. Through a large number of experiments, we present results that show the data processing limits for each algorithm in WEKA, as well as their corresponding execution times as a function of the number of records and attributes respectively.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κ. Γιάννη Θεοδωρίδη για την ανάθεση της παρούσας διπλωματικής καθώς και για την εμπιστοσύνη και την υπομονή που επέδειξε κατά την διάρκεια εκπόνησης της.

Ευχαριστώ ιδιαίτερα τους υποψήφιους διδάκτορες κα. Δέσποινα Κοπανάκη και κ. Παναγιώτη Ταμπάκη για την πολύτιμη εποπτεία, καθοδήγησή και υποστήριξή τους αλλά και για τον χρόνο που μου αφιέρωσαν.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου που με στηρίζουν σε κάθε μου προσπάθεια και ιδιαίτερα τον Δημήτρη που είναι πάντα δίπλα μου σε ότι χρειαστώ.

Κεφάλαιο 1 - Εισαγωγή

Τα τελευταία χρόνια η ανάγκη για την ανάπτυξη τεχνικών που αφορούν την αποθήκευση, την οργάνωση και την επεξεργασία διαφόρων πληροφοριών, αποτελεί ένα από τα κρισιμότερα και πιο γρήγορα αναπτυσσόμενα πεδία έρευνας στον τομέα της πληροφορικής. Είναι γεγονός ότι η πληροφορία που λαμβάνεται καθημερινά (μέσω διαδικτύου, χρηματιστήριων, επιστημονικών μελετών κτλ.) είναι περισσότερη από αυτή που μπορούμε να διαχειριστούμε. Αντιμέτωποι λοιπόν με αυτές τις μεγάλες συλλογές δεδομένων πολλοί ερευνητές σε όλο τον κόσμο προσπαθούνε να αναπτύξουνε εύρωστες, αποδοτικές και αποτελεσματικές μεθόδους για την διαχείρισή τους. Η πλειονότητα αυτών των τεχνικών βασίζεται στον ερευνητικό τομέα της Εξόρυξης Δεδομένων (*Data Mining*) [1][2][3][4].

Με τον όρο Εξόρυξη Δεδομένων αναφερόμαστε στην αποκόμιση της χρήσιμης πληροφορίας και στην εξαγωγή γνώσης από δεδομένα που είναι δύσκολο να αντιληφθούμε τις μεταξύ τους σχέσεις. Για τη διαδικασία της ευρέσεως συσχετίσεων μεταξύ των δεδομένων, χρησιμοποιούνται συνήθως τεχνικές οι οποίες βασίζονται στην ανακάλυψη μοτίβων (patterns) τα οποία συνήθως δεν είναι ευδιάκριτα [5].

Εικόνα 1.1: Εξόρυξη γνώσης από μεγάλες βάσεις δεδομένων

Γενικά οι βασικές τεχνικές της εξόρυξης δεδομένων μπορούν να κατηγοροποιηθούν σε δύο κλάσεις: α) στην περιγραφική εξόρυξη δεδομένων (Descriptive data mining) και β) στην εξόρυξη δεδομένων που έχει σκοπό την πρόβλεψη (Predictive Data Mining). Σκοπός της περιγραφικής ανάλυσης δεδομένων είναι να παρουσιάσει με ένα περιληπτικό τρόπο τις πιο σημαντικές ιδιότητες των δεδομένων. Από την άλλη πλευρά η εξόρυξη δεδομένων με σκοπό την πρόβλεψη, εστιάζει στη εξαγωγή του μηχανισμού (μοντέλο) ο οποίος είναι υπεύθυνος για την παραγωγή των δεδομένων.

Ένα σύστημα εξόρυξης δεδομένων είναι υπεύθυνο για την εκτέλεση μίας ή περισσότερων εργασιών οι οποίες περιγράφονται παρακάτω:

1. Περιγραφή κλάσης (*Class Description*): Σκοπός της διεργασίας περιγραφής κλάσης είναι να μας παρέχει μια σύντομη αλλά σαφή περίληψη της συλλογής δεδομένων που έχουμε και να τα ξεχωρίσει από άλλα. Ο περιληπτικός τρόπος έκφρασης μιας συλλογής δεδομένων καλείται χαρακτηρισμός κλάσης (*Class Characterization*). Στην περιγραφή κλάσης δεν πρέπει να παρέχονται μόνο περιληπτικές πληροφορίες όπως είναι ο μέσος όρος, το άθροισμα τιμών κτλ. αλλά και πληροφορίες των δεδομένων που αφορούν άλλα στατιστικά μεγέθη όπως είναι η διασπορά, interquartile range (IQR) κτλ.
2. Συσχέτιση (*Association*): Σκοπός του μηχανισμού συσχέτισης είναι να ανακαλυφθούν κανόνες ανάμεσα στα δεδομένα όπως για παράδειγμα η ανίχνευση τιμών οι οποίες παρουσιάζονται κατά ζεύγη σε ένα σύνολο δεδομένων.

3. Κατηγοριοποίηση (*Classification*): Σκοπός της κατηγοριοποίησης είναι η ανάλυση ενός set από δεδομένα εκπαίδευσης (*Training Data*) και η κατασκευή μοντέλων για κάθε κατηγορία, βασισμένη στα χαρακτηριστικά των δεδομένων. Κατά την εργασία αυτή παράγονται κανόνες οι οποίοι μπορούν να χρησιμοποιηθούν για την καλύτερη κατανόηση των κλάσεων μέσα σε μια βάση και για την κατηγοριοποίηση μελλοντικών δεδομένων. Για παράδειγμα η κατηγοριοποίηση μιας ασθένειας μπορεί να βοηθήσει στο να προβλεφθεί η ασθένεια βασισμένη στα συμπτώματα του ασθενή.
4. Πρόβλεψη (*Prediction*): Σκοπός αυτής της λειτουργίας είναι να προβλέπει τις πιθανές τιμές κάποιων δεδομένων που λείπουν ή να εκτιμήσει την τιμή μιας κατανομής συγκεκριμένων χαρακτηριστικών ενός συνόλου όπως για παράδειγμα τον πιθανό μισθό που παίρνει ένας υπάλληλος βασιζόμενη στη κατανομή των μισθών υπαλλήλων που έχουν παρόμοια θέση στη εταιρία. Για να γίνει αυτό συνήθως χρησιμοποιούνται μοντέλα παλινδρόμησης (*γραμμικά ή μη γραμμικά*), δέντρα απόφασης, νευρωνικά δίκτυα κτλ.
5. Συσταδοποίηση (*Clustering*): Σκοπός της συσταδοποίησης είναι να ανιχνεύσει ομάδες με βάση κάποια κοινά χαρακτηριστικά από ένα σύνολο δεδομένων. Για να γίνει αυτό χρειάζεται μια μετρική ομοιότητας η οποία εκφράζεται από μία συνάρτηση απόστασης.

Κάθε μία από τις παραπάνω διεργασίες απαιτούν τη ανάπτυξη αποδοτικών και αποτελεσματικών αλγορίθμων οι οποίοι πολύ συχνά δανείζονται από τον κλάδο της μηχανικής μάθησης. Η μηχανική μάθηση είναι το πεδίο μελέτης το οποίο δίνει στους υπολογιστές την ικανότητα να "μαθαίνουν" χωρίς να έχουν άμεσα προγραμματιστεί για αυτό [6]. Παρόλο που η μηχανική μάθηση και η εξόρυξη δεδομένων είναι δύο όροι που συχνά συγχέονται, μιας και χρησιμοποιούν τις ίδιες μεθόδους, ωστόσο παρουσιάζουν κάποιες διαφορές. Η μηχανική μάθηση εστιάζει στην πρόβλεψη, βασισμένη σε γνωστές ιδιότητες που έχουμε μάθει από τα δεδομένα εκπαίδευσης. Από την άλλη η εξόρυξη δεδομένων εστιάζει στην ανακάλυψη προηγούμενων, άγνωστων ιδιοτήτων των δεδομένων. Τέλος αξίζει να σημειωθεί ότι στη μηχανική μάθηση η απόδοση των αλγορίθμων της, συχνά αποτιμάται ως προς την ικανότητα τους να αναπαράγουν την ήδη γνωστή γνώση, ενώ στην εξόρυξη δεδομένων αποτιμώνται με βάση την ικανότητά τους να ανιχνεύσουν άγνωστη πληροφορία [7][8].

Γενικά οι αλγόριθμοι μηχανικής μάθησης, τους οποίους δανείζεται και η εξόρυξη γνώσης για τη διενέργεια των προαναφερθέντων εργασιών, μπορούν να διαχωριστούν σε δύο μεγάλες κατηγορίες: στην "Μάθηση με Επιτήρηση" (*Supervised Learning*) και στην "Μάθηση χωρίς Επιτήρηση" (*Unsupervised Learning*).

Γενικός σκοπός της Μάθησης με Επιτήρηση είναι να παράγει μια συνάρτηση (*κανόνα*) όπου αντιστοιχεί τα δεδομένα εισόδου σε επιθυμητές τιμές εξόδου. Μια εφαρμογή αυτής της μάθησης είναι σε προβλήματα κατηγοριοποίησης (*Classification*).

Σκοπός της Μάθησης χωρίς Επιτήρηση είναι να μοντελοποιήσει ένα σύνολο εισόδων σε κλάσεις. Η τεχνική αυτή χρησιμοποιείται ευρέως στην εξόρυξη δεδομένων και στην ανακάλυψη γνώσης όπου οι κατηγορίες που ανήκουν τα δεδομένα δεν είναι γνωστές εξαρχής.

Από τις παραπάνω μεθόδους εξόρυξης δεδομένων [9][10], οι πιο σημαντικές και ευρέως χρησιμοποιούμενες είναι η κατηγοριοποίηση και η συσταδοποίηση οι οποίες αναλύονται εκτενέστερα παρακάτω.

Κατηγοριοποίηση Δεδομένων

Οι αλγόριθμοι κατηγοριοποίησης έχουν ως στόχο την πρόβλεψη της κατηγορίας (class) στην οποία ανήκει μια μη κατηγοριοποιημένη παρατήρηση (observation) [11][12][13][14] . Για να επιτευχθεί αυτό, χρησιμοποιούνται δεδομένα των οποίων είναι γνωστή η κατηγορία στην οποία ανήκουν. Τα δεδομένα αυτά χρησιμοποιούνται για εκπαίδευση των κατηγοριοποιητών και γι αυτό το λόγο ονομάζονται δεδομένα εκπαίδευσης. Στόχος των κατηγοριοποιητών είναι να προβλέψουν την κατηγορία στην οποία ανήκει μια νέα παρατήρηση με την μεγαλύτερη δυνατή ακρίβεια.

Η κατηγοριοποίηση βασίζεται στην εξέταση των χαρακτηριστικών μιας νέας παρατήρησης η οποία αντιστοιχίζεται με βάση τα χαρακτηριστικά της σε ένα προκαθορισμένο σύνολο κατηγοριών ή κλάσεων. Σήμερα οι μέθοδοι κατηγοριοποίησης αποτελούν τις πιο διαδεδομένες τεχνικές στην

εξόρυξη γνώσης. Πράγματι πολλές εταιρίες, βιομηχανίες, τράπεζες, υπηρεσίες υγείας κ.α. χρησιμοποιούν σε καθημερινή βάση κατάλληλα προσαρμοσμένα συστήματα κατηγοριοποίησης.

Κάθε σύστημα κατηγοριοποίησης αναπτύσσεται για μια συγκεκριμένη εφαρμογή. Προκειμένου να αναπτύξουμε ένα αξιόπιστο σύστημα κατηγοριοποίησης θα πρέπει να έχουμε διαθέσιμο ένα επαρκές σύνολο δεδομένων εκπαίδευσης. Τα δεδομένα εκπαίδευσης αποτελούν ένα δείγμα δεδομένων όπου για κάθε δείγμα εισόδου γνωρίζουμε την τιμή εξόδου. Χρησιμοποιώντας τα δεδομένα εκπαίδευσης εφαρμόζουμε κατάλληλες τεχνικές ώστε να δημιουργήσουμε ένα μοντέλο το οποίο είναι σε θέση να κατηγοριοποιεί τα δεδομένα στον προκαθορισμένο αριθμό κλάσεων με όσο το δυνατό μεγαλύτερη ακρίβεια.

Ένα πολύ σημαντικό θέμα σχετικό με την κατηγοριοποίηση είναι η υπερπροσαρμογή (overfitting). Με τον όρο υπεπροσαρμογή εννοούμε το φαινόμενο κατά το οποίο το μοντέλο (η συνάρτηση) κατηγοριοποίησης έχει πάρα πολλές παραμέτρους και εμφανίζει μεγάλη ελαστικότητα προκειμένου να καταφέρει να κατηγοριοποιήσει σωστά όλα τα δεδομένα εκπαίδευσης. Αν και αυτά τα μοντέλα κατηγοριοποιούν με μεγάλη ακρίβεια τα δεδομένα εκπαίδευσης, παρουσιάζουν συνήθως το σημαντικό μειονέκτημα να μην κατηγοριοποιούν σωστά δεδομένα τα οποία δεν συμμετείχαν στην εκπαίδευση. Έτσι προκειμένου να αντιμετωπιστεί το πρόβλημα της υπερπροσαρμογής έχουν προταθεί αρκετές τεχνικές [2] (regularization, Occam's razor) οι οποίες ρυθμίζουν κατάλληλα την πολυπλοκότητα του μοντέλου.

Μέθοδοι εκτίμησης της απόδοσης ενός κατηγοριοποιητή

Τα περισσότερα μέτρα που χρησιμοποιούμε προκειμένου να εξετάσουμε την απόδοση ενός κατηγοριοποιητή είναι τα παρακάτω:

1. Ακρίβεια (accuracy): Η ακρίβεια κατηγοριοποίησης είναι ένα μέγεθος το οποίο μας δείχνει την ικανότητα του μοντέλου να προβλέπει την κατηγορία μιας νέας παρατήρησης, η οποία δεν έχει χρησιμοποιηθεί στη εκπαίδευση του μοντέλου.
2. Ταχύτητα (speed): Όσο αυξάνεται ο όγκος των δεδομένων, ο παράγοντας της ταχύτητας διαδραματίζει όλο και σημαντικότερο ρόλο στην αποτίμηση της απόδοσης του αλγορίθμου. Εκτός από το χρόνο εκτέλεσης του αλγορίθμου κατηγοριοποίησης ο οποίος είναι άμεσα συνδεδεμένος με το μέγεθος και τη διαστατικότητα των δεδομένων προς κατηγοριοποίηση, σημαντικό ρόλο διαδραματίζει και η ταχύτητα εκπαίδευσης του κατηγοριοποιητή προκειμένου να καθοριστεί το μοντέλο κατηγοριοποίησης.
3. Ευρωστία (robustness): Με αυτόν τον όρο αποτιμούμε πόσο ανεκτικός είναι ο αλγόριθμος δηλαδή πόσο σωστά κατηγοριοποιεί δεδομένα τα οποία μπορεί να είναι ελλιπή ή να έχουν αλλοιωθεί από θόρυβο.
4. Κλιμάκωση (scalability): Με τον όρο αυτό αποτιμούμε πόσο αποδοτική είναι η κατασκευή ενός μοντέλου κατηγοριοποίησης δοθέντος μεγάλου αριθμού δεδομένων εκπαίδευσης.
5. Διερμηνευτικότητα (interpretability): Με αυτό τον όρο αποτιμούμε το επίπεδο κατανόησης και γνώσης που παρέχεται από το μοντέλο. (Μπορεί να εκτιμηθεί μετρώντας το πόσο πολύπλοκο είναι το μοντέλο π.χ. Αριθμός κόμβων στα δέντρα απόφασης, αριθμός επιπέδων στα νευρωνικά δίκτυα κ.α.).

Επιπλέον, ένας κατηγοριοποιητής μπορεί να αξιολογηθεί βάσει του κατά πόσο μπορεί να διακρίνει με ακρίβεια κλάσεις οι οποίες δεν αποτελούνται από μικρό αριθμό παρατηρήσεων οι οποίοι συνήθως λαμβάνονται ως ακραίες παρατηρήσεις (outliers). Αυτή η ικανότητα διάκρισης αποτελεί ένα πολύ βασικό χαρακτηριστικό σε περιπτώσεις όπου μας ενδιαφέρει να εστιάσουμε στις ακραίες αυτές παρατηρήσεις. Για παράδειγμα είναι σημαντικότερο ο αλγόριθμός μας να μπορεί να προβλέψει με μεγάλη ακρίβεια ένα ακραίο καιρικό φαινόμενο (τυφώνας, χιονοθύελλα κ.τ.λ.) που μπορεί να στοιχίσει ανθρώπινες ζωές από το να προβλέψει εσφαλμένα αν θα βρέξει αύριο.

Συσταδοποίηση Δεδομένων

Με τον όρο συσταδοποίηση (clustering) δεδομένων εννοούμε την διαδικασία η οποία κατατάσσει τα δεδομένα σε ομάδες (συστάδες - clusters), βρίσκοντας ομοιότητες μεταξύ των δεδομένων βάσει των χαρακτηριστικών που υπάρχουν σε αυτά. Η συσταδοποίηση είναι μία από τις πιο γνωστές και

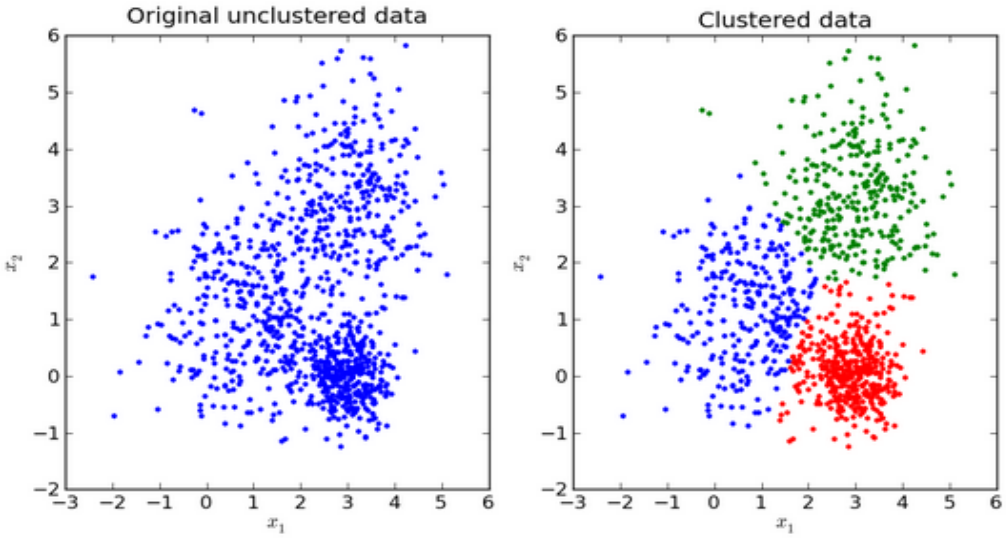
δημοφιλής τεχνικές εξόρυξης γνώσης. Είναι παρόμοια με την κατηγοριοποίηση καθώς και στις δύο περιπτώσεις τα δεδομένα οργανώνονται σε ομάδες. Η διαφορά της κατηγοριοποίησης σε σχέση με την συσταδοποίηση δεδομένων είναι ότι, στην κατηγοριοποίηση οι ομάδες στις οποίες θα ενταχθούν τα δεδομένα, είναι προκαθορισμένες που σημαίνει ότι οι ομάδες είναι εξαρχής γνωστές. Αντιθέτως στην συσταδοποίηση, οι ομάδες δεν είναι πάντα καθορισμένες αλλά δημιουργούνται δυναμικά με βάση την ομοιογένεια των δεδομένων. "*Για την δημιουργία μιας συστάδας (cluster) πρέπει η απόσταση μεταξύ των σημείων κάποιας συστάδας να είναι μικρότερη από την απόσταση μεταξύ ενός σημείου της συστάδας και οποιουδήποτε σημείου εκτός συστάδας" [15].*

Η τεχνική της συσταδοποίησης βρίσκει εφαρμογή σε πολλούς τομείς οι οποίοι απαιτούν την λήψη αποφάσεων. Σημαντικές εφαρμογές εμφανίζονται στην πληροφορική, την ιατρική, την βιολογία, την οικονομία και σε άλλα επιστημονικά πεδία. Με την εφαρμογή της συσταδοποίησης σε πραγματικές βάσεις δεδομένων προκύπτουν πολλά ενδιαφέροντα προβλήματα.

Ακραία σημεία (Outliers): Ένα βασικό πρόβλημα της συσταδοποίησης είναι ο χειρισμός των ακραίων σημείων. Τα σημεία αυτά δεν μπορούν να ενταχθούν σε καμία συστάδα. Στην περίπτωση όμως που ενταχθούν σε κάποια συστάδα (εξαρτάται από την επιλογή του αλγορίθμου, την επιλογή του αριθμού των ομάδων και άλλων κριτηρίων) τότε η συσταδοποίηση ενδεχομένως να μην δώσει τα επιθυμητά αποτελέσματα.

Δυναμικά δεδομένα: Μία συστάδα δεδομένων μπορεί να αλλάξει σύσταση όταν σε μία βάση δεδομένων περιλαμβάνονται δυναμικά δεδομένα. Κάτι που είναι αναπόφευκτο σε μία πραγματική βάση δεδομένων.

Προσδιορισμός συστάδας: Κατά την διαδικασία της συσταδοποίησης δημιουργούνται συστάδες οι οποίες δεν είναι πάντα δυνατόν να ερμηνευτούν. Σε αυτή την περίπτωση είναι απαραίτητη η ανάλυση των δεδομένων από ειδικούς ώστε να δοθούν ετικέτες και να προσδιοριστεί η σημασία της κάθε συστάδας. Επιπλέον αξίζει να σημειωθεί ότι σε ένα πρόβλημα συσταδοποίησης δεν υπάρχει μία μόνο σωστή λύση. Είναι πολύ δύσκολο, ειδικά σε μεγάλα αρχεία δεδομένων, να προσδιοριστεί ο ακριβής αριθμός συστάδων.



Εικόνα 1.2: Συσταδοποίηση δεδομένων με τρεις κλάσεις. Στο σχήμα αριστερά φαίνονται τα δεδομένα στις δύο διαστάσεις. Στο σχήμα δεξιά παρουσιάζονται τα δεδομένα μετά τον διαχωρισμό τους σε συστάδες.

Ταξινόμηση των αλγορίθμων συσταδοποίησης

Στη διεθνή βιβλιογραφία έχουν προταθεί αρκετοί τρόποι ταξινόμησης των αλγορίθμων συσταδοποίησης [16][17][18][19][20]. Οι επικρατέστερες ομάδες ταξινόμησης των αλγορίθμων συσταδοποίησης περιγράφονται παρακάτω:

Ιεραρχικοί αλγόριθμοι

Σκοπός των ιεραρχικών αλγορίθμων είναι η δημιουργία ιεραρχίας στις δημιουργούμενες συστάδες. Οι αλγόριθμοι ιεραρχικής συσταδοποίησης ορίζονται σε δύο κατηγορίες:

1. Συσσωρευτικοί - Προσέγγιση από κάτω προς τα πάνω (Agglomerative - bottom up approach): Σε αυτή την κατηγορία οι αλγόριθμοι ξεκινούν με την υπόθεση ότι κάθε παρατήρηση (observation) είναι μια ξεχωριστή ομάδα. Στη συνέχεια χρησιμοποιώντας κατάλληλες μετρικές συγχωνεύει διαδοχικά τις ομάδες που θεωρείται ότι έχουν την μεγαλύτερη ομοιότητα μεταξύ τους.
2. Διαιρετικοί - Προσέγγιση από πάνω προς τα κάτω (Divisive - top down approach): Σε αυτή την κατηγορία όλες οι παρατηρήσεις ξεκινούν σαν μία ομάδα. Στη συνέχεια χρησιμοποιώντας κατάλληλες μετρικές, οι παρατηρήσεις αυτές διαιρούνται διαδοχικά σε μικρότερες ομάδες.

Και στις δύο περιπτώσεις οι συγχωνεύσεις και διαιρέσεις των παρατηρήσεων αντίστοιχα, παρουσιάζονται με την μορφή δενδροδιαγράμματος. Το βασικότερο μειονέκτημα των ιεραρχικών αλγορίθμων είναι η μεγάλη υπολογιστική πολυπλοκότητα τους, γεγονός που τους κάνει πολύ αργούς όταν έχουν να εργαστούν με μεγάλους όγκους δεδομένων υψηλών διαστάσεων.

Διαμεριστικοί Αλγόριθμοι (Partitioning algorithms)

Η οικογένεια των διαμεριστικών αλγορίθμων μπορεί να χωριστεί σε δύο κατηγορίες: α) στους k-μέσων (k-means) και β) στους k-medoids. Η βασική διαφορά μεταξύ αυτών των δύο κατηγοριών είναι η εξής:

Στους k-means αλγορίθμους κάθε συστάδα αναπαρίσταται από το κέντρο βάρους των παρατηρήσεων που ανήκουν σε αυτή, το οποίο καλείται κεντροειδές (centroid). Σε αντίθεση στους k-medoid αλγορίθμους, η ομάδα αναπαρίσταται από την παρατήρηση που βρίσκεται πιο κεντρικά στην ομάδα το οποίο καλείται medoid.

Αν και οι διαχωριστικοί αλγόριθμοι έχουν πολύ χαμηλότερη πολυπλοκότητα από τους ιεραρχικούς με αποτέλεσμα να υπερτερούν σημαντικά σε ταχύτητα εκτέλεσης, έχουν ένα πολύ σημαντικό μειονέκτημα το οποίο είναι η ανάγκη της εκ των προτέρων καθορισμού του αριθμού των συστάδων στις οποίες θα ομαδοποιηθούν τα δεδομένα. Η ανάγκη καθορισμού αυτού του αριθμού αποτελεί σημαντικότατο μειονέκτημα αυτών των μεθόδων διότι: α) σε πολλές εφαρμογές ο αριθμός των συστάδων είναι άγνωστος και δεν μπορεί εύκολα να εκτιμηθεί εκ των προτέρων, β) μια λάθος εκτίμηση του αριθμού των συστάδων μπορεί να μας οδηγήσει σε αποτελέσματα τα οποία να μην αναδεικνύουν τις σημαντικές πτυχές των δεδομένων. Παρόλα αυτά τις τελευταίες δεκαετίες έχουν γίνει σημαντικές μελέτες πάνω στο κομμάτι του βέλτιστου καθορισμού του αριθμού των συστάδων γεγονός που κάνει την χρήση των διαχωριστικών αλγορίθμων όλο και πιο δελεαστική.

Αλγόριθμοι Ταξινόμησης βάσει της πυκνότητας (Density Based Algorithms)

Σε αντίθεση με τις περισσότερες διαχωριστικές μεθόδους, οι density-based αλγόριθμοι [21] [22] χρησιμοποιούν συναρτήσεις πυκνότητας για να αναγνωρίσουν τις συστάδες. Οι αλγόριθμοι αυτοί αναγνωρίζουν τις συστάδες με βάση την πυκνότητα των παρατηρήσεων στους χώρους που ορίζονται. Τα βασικότερα πλεονεκτήματα αυτών των αλγορίθμων είναι ότι:

α) οι συστάδες που δημιουργούνται μπορεί να έχουν οποιοδήποτε σχήμα και β) είναι ικανοί να αναγνωρίζουν τις παρατηρήσεις που αντιστοιχούν σε ακραίες τιμές (outliers).

Αλγόριθμοι βασισμένοι σε πλέγμα (Grid Based Algorithms)

Οι αλγόριθμοι συσταδοποίησης βασισμένοι σε πλέγμα (grid-based) [23] αποτελούν μια οικογένεια αλγορίθμων η οποία τα τελευταία χρόνια κερδίζει συνεχώς έδαφος έναντι των άλλων κατηγοριών συσταδοποίησης, λόγω του γρήγορου χρόνου εκτέλεσης τους. Ο γρήγορος χρόνος εκτέλεσης αυτών των αλγορίθμων οφείλεται κυρίως στο μέγεθος του πλέγματος που χρησιμοποιείται και όχι στον αριθμό των παρατηρήσεων προς συσταδοποίηση. Αυτές οι μέθοδοι χρησιμοποιούν ένα

πλέγμα το οποίο διαχωρίζει τον χώρο, που "ζουν" οι παρατηρήσεις μας, σε κελιά. Κάθε παρατήρηση αντιστοιχίζεται σε ένα κελί και αναπαριστάται από ένα σύνολο στατιστικών παραμέτρων οι οποίες εκτιμώνται από τις παρατηρήσεις που ανήκουν στο συγκεκριμένο κελί. Με αυτό τον τρόπο η συσταδοποίηση μπορεί να εκτελεστεί χρησιμοποιώντας την πληροφορία των κελιών του πλέγματος αντί για της κάθε αυτού παρατηρήσεις. Λόγω του ότι το μέγεθος του πλέγματος είναι συνήθως πολύ μικρότερο από τον αριθμό των παρατηρήσεων, η ταχύτητα εκτέλεσης του αλγορίθμου μπορεί να βελτιωθεί σημαντικά. Ωστόσο το πλεονέκτημα αυτών των αλγόριθμων αναιρείται, όταν οι παρατηρήσεις παρουσιάζουν μεγάλη ανομοιογένεια, μιας και σε αυτές τις περιπτώσεις χρειαζόμαστε ένα πολύ πυκνό πλέγμα για να μπορέσουμε να έχουμε καλά αποτελέσματα συσταδοποίησης.

Τα τελευταία χρόνια η ερευνητική κοινότητα έχει στρέψει τις προσπάθειες της στον να μπορέσει να βρει τρόπο αντιμετώπισης αυτού του προβλήματος. Η πιο διαδεδομένη μέθοδος που χρησιμοποιούν είναι να χρησιμοποιούν πλέγμα πολλαπλής πυκνότητας το οποίο είναι πυκνότερο (αραιότερο) σε περιοχές όπου η ανομοιογένεια των παρατηρήσεων είναι μεγάλη (μικρή).

Άλλοι τύποι αλγορίθμων συσταδοποίησης

Εκτός από τις παραπάνω βασικές κατηγορίες αλγορίθμων συσταδοποίησης οι οποίες διαχωρίζουν το χώρο των παρατηρήσεων σε συστάδες έτσι ώστε κάθε παρατήρηση να ανήκει σε μία και μόνο συστάδα, υπάρχουν και αλγόριθμοι οι οποίοι δεν ικανοποιούν αυτή την υπόθεση όπως είναι οι αλγόριθμοι ασαφούς συσταδοποίησης, τα νευρωνικά δίκτυα και οι γενετικοί αλγόριθμοι [24].

Τα βασικά στάδια της συσταδοποίησης

Τα βασικά βήματα της διαδικασίας της συσταδοποίησης είναι:

1. Εξαγωγή χαρακτηριστικών από τα δεδομένα.
2. Επιλογή μετρικής για τον υπολογισμό των αποστάσεων μεταξύ των παρατηρήσεων.
3. Συσταδοποίηση.
4. Περιγραφή συστάδων (όπου χρειάζεται).
5. Αποτίμηση της συσταδοποίησης.

Στο πρώτο βήμα γίνεται μια προεπεξεργασία των δεδομένων έτσι ώστε αν επιλεχτούν τα πιο σημαντικά χαρακτηριστικά για την περαιτέρω ανάλυση. Η προεπεξεργασία αυτή μπορεί να συμπεριλαμβάνει τεχνικές όπως ανάκτηση χαμένων τιμών, μείωση της διαστατικότητας των δεδομένων κ.α.

Στο δεύτερο βήμα γίνεται η επιλογή της μετρικής απόστασης. Η πιο συχνά επιλεγόμενη μετρική είναι η Ευκλείδεια απόσταση. Παρόλα αυτά στη διεθνή βιβλιογραφία υπάρχει ένας μεγάλος αριθμός μετρικών των οποίων η χρήση εξαρτάται από το πρόβλημα. Κάποιες από τις πιο ευρέως χρησιμοποιούμενες συναρτήσεις απόστασης παρουσιάζονται στον παρακάτω πίνακα.

Πίνακας συναρτήσεων απόστασης	
Euclidean	$d = \sqrt{\sum_{i=1}^{n} \Sigma (x_i - y_i)^2}$
Manhattan	$d = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Pearson Correlation and Pearson Squared	$d = 1 - r$ $r = Z(x) \bullet Z(y) / n$
Chebychev	$Maxi \lvert X_i - Y_i \rvert$
Spearman Rank Correlation	$1 - \frac{6\sum_{i=1}^{n} (rank(X_i)) - rank(Y_i))^2}{n(n^2 - 1)}$

Πίνακας 1.1: Συναρτήσεις απόστασης

Στο τρίτο βήμα εφαρμόζεται κατ' επιλογήν ο αλγόριθμος συσταδοποίησης. Στο τέταρτο βήμα εφαρμόζονται μέθοδοι οι οποίες μας επιτρέπουν να περιγράψουμε με σύντομο και απλό τρόπο τις συστάδες που δημιουργήθηκαν από την εκτέλεση του αλγορίθμου. Η αναπαράσταση αυτή γίνεται συνήθως με την επιλογή κάποιου ή κάποιων αντιπροσωπευτικών σημείων της ομάδας είτε με την εξαγωγή στατιστικών χαρακτηριστικών από την κάθε συστάδα. Η σύντομη αναπαράσταση των συστάδων μας βοηθά τόσο στο να αντιληφθούμε εύκολα και γρήγορα τα χαρακτηριστικά των συστάδων όσο και να προσθέσουμε σε κάθε συστάδα νέες παρατηρήσεις που μπορεί να γίνουν διαθέσιμες.

Στο πέμπτο βήμα γίνεται η αποτίμηση των αποτελεσμάτων. Αξίζει να σημειωθεί ότι τα αποτελέσματα της συσταδοποίησης μπορεί να διαφέρουν σημαντικά τόσο μεταξύ διαφορετικών αλγορίθμων συσταδοποίησης όσο και μεταξύ διαφορετικών παραμετροποιήσεων του ίδιου αλγορίθμου. Παρόλες τις μεθόδους που έχουν προταθεί στη διεθνή βιβλιογραφία για την αποτίμηση της ποιότητας συσταδοποίησης των αλγορίθμων, δεν υπάρχουν γενικές μέθοδοι αποτίμησης τους μιας και η ποιότητα της συσταδοποίησης σχετίζεται άμεσα με τη φύση του προβλήματος.

Σκοπός Εργασίας

Βασικός στόχος της παρούσας διπλωματικής εργασίας είναι να μελετηθούν και να συγκριθούν οι βασικότερες μέθοδοι συσταδοποίησης ως προς την δυνατότητά τους να διαχειρίζονται μεγάλα αρχεία δεδομένων με την χρήση του δημοφιλούς λογισμικού WEKA. Το λογισμικό WEKA το οποίο αναπτύχθηκε από το πανεπιστήμιο Waikato της Νέας Ζηλανδίας, αποτελεί μία από τις πιο ευρέως διαδεδομένες πλατφόρμες οι οποίες χρησιμοποιούνται από επιστήμονες διαφόρων ειδικοτήτων με σκοπό την ανάλυση δεδομένων για την εξαγωγή συμπερασμάτων. Τα τελευταία χρόνια έχουν αναπτυχθεί πολλά πακέτα λογισμικού με κύριο σκοπό την ανάλυση δεδομένων. Οι βασικοί λόγοι για τους οποίους προτιμάται το λογισμικό WEKA είναι διότι διατίθεται δωρεάν, είναι επεκτάσιμο και ιδιαίτερα φιλικό προς τον χρήστη, καθώς περιέχει έτοιμους προς εκτέλεση τους σημαντικότερους αλγορίθμους συσταδοποίησης και κατηγοριοποίησης γεγονός που διευκολύνει σημαντικά την διαδικασία ανάλυσης δεδομένων.

Η δομή της παρούσας διπλωματικής παρουσιάζεται παρακάτω:

Στο **Κεφάλαιο 2** γίνεται εκτενής παρουσίαση του συστήματος WEKA. Πιο συγκεκριμένα γίνεται περιγραφή των βασικών λειτουργιών του, παρουσιάζονται οι οθόνες και τα μενού επιλογής του λογισμικού, ο τρόπος εκτέλεσης αλγορίθμων καθώς και οι οθόνες αποτελεσμάτων και διαγραμμάτων οπτικοποίησής τους.

Στο **Κεφάλαιο 3** περιγράφονται οι αλγόριθμοι συσταδοποίησης που περιέχονται στο σύστημα WEKA. Επίσης παρουσιάζονται οι παράμετροι εισόδου όπως απαιτούνται από το σύστημα WEKA καθώς και τα αποτελέσματα εξόδου.

Στο **Κεφάλαιο 4** παρουσιάζουμε την ικανότητα του κάθε αλγορίθμου να διαχειριστεί μεγάλα αρχεία δεδομένων μέσα από το σύστημα WEKA. Πιο συγκεκριμένα δίνουμε τα αποτελέσματα μιας σειράς πειραμάτων τα οποία δείχνουν τα όρια και τους χρόνους εκτέλεσης του κάθε αλγορίθμου καθώς αυξάνονται ο αριθμός χαρακτηριστικών και εγγραφών ενός αρχείου δεδομένων.

Στο **Κεφάλαιο 5** περιγράφεται ο αλγόριθμος CURE ο οποίος δεν περιέχεται στη λίστα αλγορίθμων του WEKA. Θεωρείται όμως ένας από τους πιο πολλά υποσχόμενους αλγορίθμους στην επεξεργασία μεγάλων αρχείων δεδομένων. Προκειμένου να δοκιμάσουμε την απόδοση του όσο αφορά την ικανότητά του να αναλύσει μεγάλα αρχεία δεδομένων, τον προγραμματίσαμε σε γλώσσα προγραμματισμού JAVA (όπως είναι υλοποιημένοι και οι αλγόριθμοι του συστήματος WEKA). Επίσης παρουσιάζουμε τα αποτελέσματα της εκτέλεσης μιας σειράς πειραμάτων, τα οποία δείχνουν την ταχύτητα εκτέλεσης του αλγορίθμου για διαφορετικού μεγέθους αρχεία δεδομένων, την ικανότητα του να διαχειρίζεται μεγάλα αρχεία δεδομένων και να αναγνωρίζει χωρικά ανομοιόμορφες συστάδες.

Τέλος στο **Κεφάλαιο 6** παρουσιάζονται τα συμπεράσματα της παρούσας διπλωματικής καθώς και πιθανές μελλοντικές επεκτάσεις οι οποίες παρουσιάζουν ιδιαίτερο ερευνητικό ενδιαφέρον.

Κεφάλαιο 2 - Περιγραφή του Λογισμικού WEKA

Τα τελευταία χρόνια έχουν αναπτυχθεί αρκετά πακέτα λογισμικού τα οποία έχουν σκοπό να βοηθήσουν τον χρήστη με απλό και φιλικό τρόπο στην ανάλυση δεδομένων. Ανάμεσα στα πιο σημαντικά πακέτα λογισμικού τα οποία είναι διαθέσιμα δωρεάν στο διαδίκτυο είναι τα RapidMiner, Tanagra, Orange και KNIME και WEKA [26] [27]. Όλα αυτά τα εργαλεία περιέχουν ένα σύνολο από αλγορίθμους μηχανικής μάθησης τους οποίους μπορεί να εφαρμόσει ο χρήστης στο αρχείο δεδομένων που τον ενδιαφέρει. Για τους σκοπούς της παρούσας εργασίας θα χρησιμοποιήσουμε το εργαλείο WEKA.

2.1 Εισαγωγή στο σύστημα WEKA

Τα αρχικά WEKA προέρχονται από τις λέξεις Waikato environment for knowledge analysis. Επίσης το όνομα που σχηματίζεται από το ακρωνύμιο του λογισμικού, αντιστοιχεί στο όνομα ενός πτηνού που ζει αποκλειστικά στην Νέα Ζηλανδία και το οποίο αποτελεί το σήμα κατατεθέν του. Το σύστημα WEKA αναπτύχθηκε από το πανεπιστήμιο Waikato της Νέας Ζηλανδίας.

Το σύστημα WEKA είναι ένα δωρεάν λογισμικό ανάλυσης το οποίο περιέχει κλασικούς αλγορίθμους μηχανικής μάθησης οι οποίοι χρησιμοποιούνται ευρέως σε εφαρμογές ανάλυσης και εξόρυξης δεδομένων. Πιο συγκεκριμένα το WEKA μας παρέχει αλγορίθμους κατηγοριοποίησης και συσταδοποίησης καθώς επίσης και μεθόδους οι οποίες μας επιτρέπουν να επεξεργαστούμε τα δεδομένα προτού τα εισάγουμε ως είσοδο στους παραπάνω αλγορίθμους. Κατά την εκτέλεση των αλγορίθμων, το WEKA μας παρέχει οπτική και ποσοτική πληροφορία η οποία μας βοηθά να αποτιμήσουμε τα αποτελέσματα του κάθε αλγορίθμου.

Το WEKA έχει αναπτυχθεί σε γλώσσα προγραμματισμού JAVA με αποτέλεσμα να είναι συμβατό με όλα τα λειτουργικά συστήματα (Windows, Linux κτλ.). Αξίζει να σημειωθεί ότι τον Αύγουστο του 2005 η ομάδα ανάπτυξης του συστήματος WEKA κέρδισε στο 11ο ACM SIGKDD διεθνές συνέδριο, το σημαντικό βραβείο υπηρεσίας στο τομέα της εξόρυξης δεδομένων και ανακάλυψης γνώσης (knowledge discovery). Έτσι το σύστημα WEKA είναι ευρέως αναγνωρισμένο από την διεθνή κοινότητα και αποτελεί ένα σημείο αναφοράς στην ιστορία της εξόρυξης δεδομένων και της μηχανικής μάθησης γενικότερα. Τέλος στα έντεκα χρόνια ιστορίας του έχει αναγνωριστεί ως ένα από τα πιο κατανοητά εργαλεία ανάλυσης δεδομένων στον κόσμο και λόγω του φιλικού περιβάλλοντος του, χρησιμοποιείται από διάφορους επιστημονικούς χώρους (βιολογίας, ιατρικής, κτλ.).

2.2 Χαρακτηριστικά συστήματος WEKA

Τα σπουδαιότερα χαρακτηριστικά του συστήματος περιγράφονται παρακάτω:

1. Το WEKA υποστηρίζει όλες τις εκδόσεις Windows και Unix καθώς και πολλά άλλα λειτουργικά συστήματα.
2. Το WEKA υποστηρίζει την δομή πολλών αρχείων κειμένου και παρέχει δυνατότητα σύνδεσης με βάσεις δεδομένων.
3. Το WEKA μπορεί να διαχειριστεί τύπους δεδομένων οι οποίες περιέχουν συνεχείς και διακριτές μεταβλητές καθώς και άλλους τύπους δεδομένων.
4. Το WEKA μας παρέχει διάφορους τρόπους προ επεξεργασίας δεδομένων όπως διαχείριση αρχείων στα οποία λείπουν τιμές (missing values), μεθόδους απαλοιφής θορύβου στις τιμές των δεδομένων, κανονικοποίηση των τιμών, ανάμιξη των δειγμάτων, διαχώριση δεδομένων, επιλογή συγκεκριμένου αριθμού χαρακτηριστικών.
5. Το WEKA μπορεί να εφαρμόσει τεχνικές αλγορίθμων κατηγοριοποίησης, συσταδοποίησης και συσχέτισης δεδομένων.

6. Το WEKA έχει την δυνατότητα οπτικοποίησης των δεδομένων και των αποτελεσμάτων κατά την κατηγοριοποίηση και συσταδοποίηση τους.
7. Το WEKA υποστηρίζει αρκετούς αλγορίθμους μηχανικής μάθησης και νευρωνικών δικτύων.
8. Το WEKA έχει την δυνατότητα παραγωγής βασικών αναφορών κατά την σύγκριση αλγορίθμων οι οποίες μας δίνουν ποσοτικά αποτελέσματα που παράγονται κατά την εκτέλεση των αλγορίθμων.

Το σύστημα του WEKA υποστηρίζει τρεις τρόπους εισαγωγής αρχείων:

a. από αρχεία (τύπος αρχείων .csv, .arff κτλ.)
b. από βάση δεδομένων
c. από διεύθυνση Url

Το WEKA έχει την δυνατότητα εισαγωγής αρχείων διαφόρων τύπων αλλά ο βασικός τύπος δεδομένων που χρησιμοποιεί είναι τύπου ARFF (Attribute Relation File Format), ο οποίος είναι ένα αρχείο κειμένου κωδικοποιημένο σε ASCII μορφή. Κάθε αρχείο ARFF μπορεί να χωριστεί σε δύο τμήματα. Το πρώτο τμήμα αποτελείται από την κεφαλίδα, η οποία περιέχει πληροφορίες σχετικά με τις δηλώσεις του ονόματος του αρχείου δεδομένων, των χαρακτηριστικών που περιέχει και των τύπων δεδομένων τους. Στο δεύτερο τμήμα περιγράφονται τα δεδομένα. Οι τύποι δεδομένων που υποστηρίζει το WEKA είναι οι εξής:

- Αριθμητικά δεδομένα (numeric)
- Ονομαστικά δεδομένα (nominal)
- Αλφαριθμητικά δεδομένα (string)
- Ημερομηνίες (date)

Αριθμητικά Χαρακτηριστικά - Numeric Attributes

Τα χαρακτηριστικά αυτά παίρνουν είτε πραγματικές τιμές είτε ακέραιες. Ο ορισμός αυτού του τύπου έχει την παρακάτω μορφή:

@attribute temperature numeric

Ονομαστικά Χαρακτηριστικά - Nominal Attributes

Τα χαρακτηριστικά αυτά παίρνουν ονομαστικές τιμές. Όλες οι δυνατές τιμές δίνονται σαν λίστα στην δήλωση των ιδιοτήτων. Ο ορισμός αυτού του τύπου έχει την παρακάτω μορφή:

@attribute windy {TRUE, FALSE}

Αλφαριθμητικά Χαρακτηριστικά - String Attributes

Οι τιμές των χαρακτηριστικών αυτών μπορεί να είναι οποιουδήποτε συνόλου χαρακτήρων, και είναι ιδιαίτερα χρήσιμα σε εφαρμογές text-mining. Ο ορισμός αυτού του τύπου έχει την παρακάτω μορφή:

@attribute Text string

Ημερομηνίες - Date Attributes

Οι τιμές των χαρακτηριστικών αυτών είναι ημερομηνίες με προαιρετικό καθορισμό μορφοποίησης σε συγκεκριμένη μορφή. Ο ορισμός αυτού του τύπου έχει την παρακάτω μορφή:

@attribute <name> date

Ωστόσο το WEKA εκτός από αρχεία τύπου ARFF, είναι συμβατό και με άλλους τύπους αρχείων όπως CSV, C4.5, binary.

2.3 Παρουσίαση του γραφικού περιβάλλοντος WEKA

Το WEKA χρησιμοποιεί μια σειρά από κλασικές τεχνικές μηχανικής μάθησης οι οποίες είναι ενοποιημένες σε ένα γραφικό περιβάλλον (Graphical User Interface - GUI) φιλικό προς το χρήστη και παρέχει εργαλεία για οπτική παρουσίαση των δεδομένων , προ επεξεργασία και ανάλυση

αποτελεσμάτων. Ξεκινώντας το WEKA εμφανίζεται το παράθυρο WEKA GUI Chooser (βλέπε εικόνα 2.1), στο οποίο ο χρήστης έχει τις εξής επιλογές:

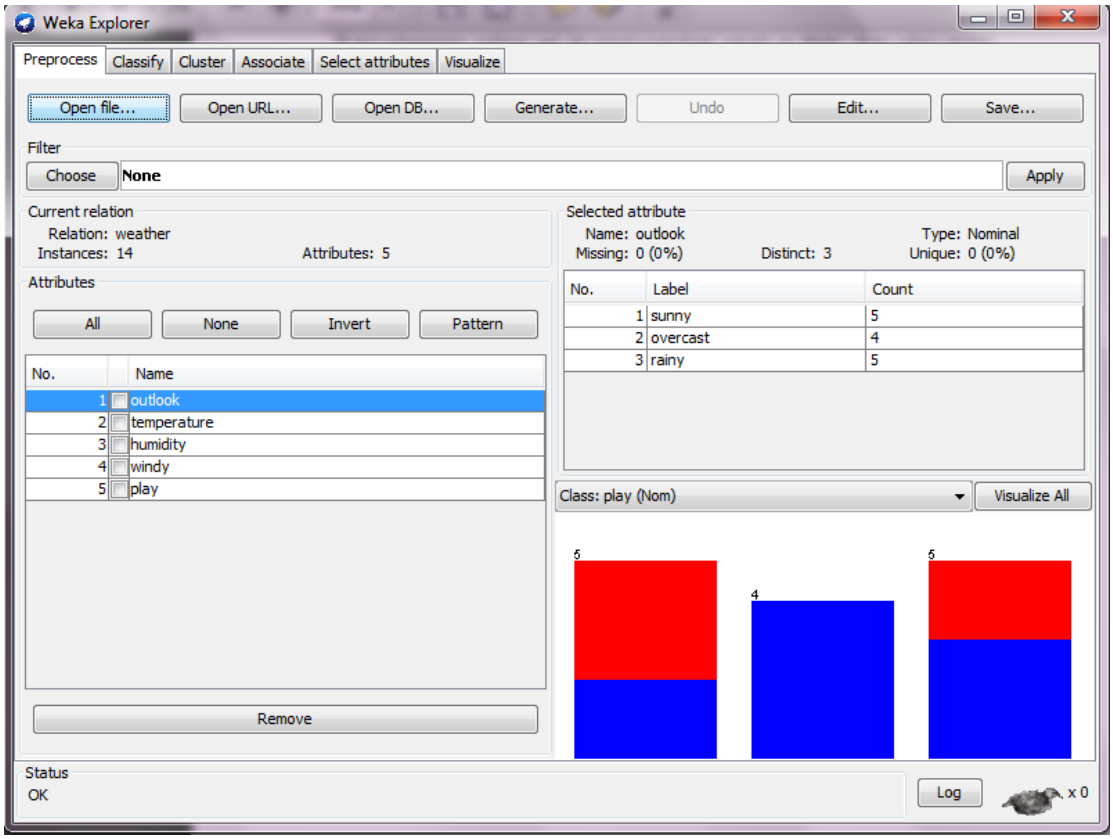
1. Explorer - Γραφικό περιβάλλον για τις ρουτίνες του WEKA και συστατικά μέρη.
2. Experimenter - Περιβάλλον εκτέλεσης πειραμάτων και στατιστικών συγκρίσεων ανάμεσα στους διάφορους αλγορίθμους.
3. Knowledge Flow - Περιβάλλον ροής γνώσης και δυνατότητα προσομοίωσης ενός συστήματος λήψης απόφασης.
4. Simple CLI (Command Line Interface) - Απευθείας εκτέλεση εντολών του WEKA από περιβάλλον γραμμής εντολών.

Εικόνα 2.1: Γραφικό περιβάλλον του WEKA - GUI Chooser

2.3.1 Explorer

Το περιβάλλον αυτό είναι το πιο εύχρηστο και φιλικό προς τον χρήστη. Μέσα από αυτό μπορούν να εφαρμοστούν όλες οι βασικές λειτουργίες του εργαλείου WEKA. Ο Explorer (βλέπε εικόνα 2.2) δίνει την δυνατότητα στον χρήστη, της προ επεξεργασίας των δεδομένων, της εφαρμογής αλγορίθμων μηχανικής μάθησης και του ελέγχου της απόδοσης αυτών. Αναλυτικότερα στο βασικό μενού του υπάρχουν οι παρακάτω επιλογές:

- Preprocess: Μέσα από αυτή την καρτέλα γίνεται επιλογή των δεδομένων είτε από αρχείο από τον τοπικό δίσκο είτε από κάποια βάση δεδομένων είτε από URL. Επιπλέον μπορεί να γίνει τροποποίηση των δεδομένων με την χρήση φίλτρων που διαθέτει και με την αφαίρεση χαρακτηριστικών.
- Classify: Σε αυτή την καρτέλα περιέχονται οι αλγόριθμοι για κατηγοριοποίηση των δεδομένων [28][29].
- Cluster: Σε αυτή την καρτέλα περιέχονται οι αλγόριθμοι για ομαδοποίηση των δεδομένων [30].
- Associate: Σε αυτή την καρτέλα περιέχονται αλγόριθμοι για την εύρεση κανόνων συσχέτισης για τα δεδομένα [31].
- Select Attributes: Στην καρτέλα αυτή γίνεται επιλογή των πιο σχετικών χαρακτηριστικών μέσα στο σύνολο των δεδομένων.
- Visualize: Μέσα από αυτή την καρτέλα γίνεται οπτικοποίηση των δεδομένων με δισδιάστατα γραφήματα.



Εικόνα 2.2: Γραφικό περιβάλλον του WEKA - Explorer

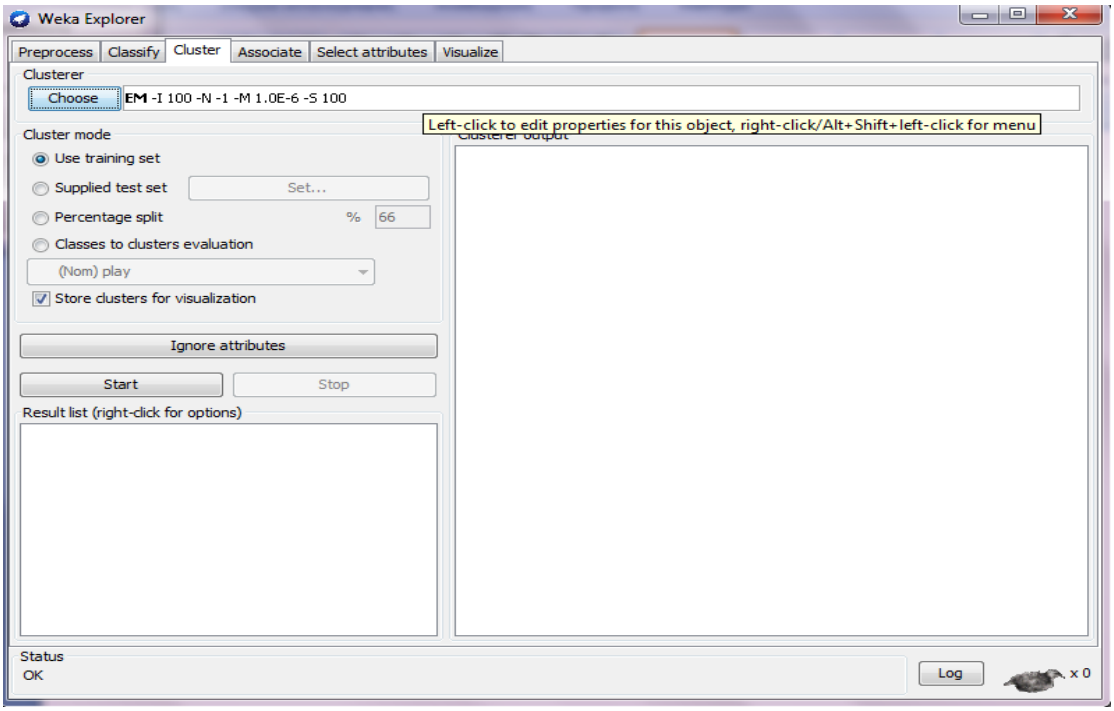
Κατά την εκκίνηση του Explorer είναι ενεργοποιημένη μόνο η πρώτη καρτέλα Preprocess. Με την επιλογή του αρχείου δεδομένων ενεργοποιούνται όλες οι υπόλοιπες καρτέλες και παράλληλα εμφανίζονται πληροφορίες για αυτό. Συγκεκριμένα εμφανίζονται το όνομα του αρχείου, ο αριθμός των εγγραφών του, ο αριθμός των χαρακτηριστικών του καθώς και το όνομα και ο τύπος όλων των χαρακτηριστικών του αρχείου αναλυτικά.

Επιπλέον σε όλες τις καρτέλες του περιβάλλοντος του Explorer εμφανίζονται το πεδίο Status και το κουμπί Log. Το πεδίο Status δείχνει μηνύματα για την κατάσταση που επικρατεί κάθε στιγμή. Επίσης με δεξί κλικ στην επιφάνεια του, μας παρέχει πληροφορίες για την διαθέσιμη μνήμη του λογισμικού καθώς και δυνατότητα απελευθέρωσης δεσμευμένης μνήμης που δεν χρησιμοποιείται για βελτίωση της απόδοσης του λογισμικού. Με το κουμπί Log ανοίγει νέο παράθυρο που παρέχει πληροφορίες κειμένου όπου σε κάθε σειρά δηλώνει την ημερομηνία και το γεγονός που έχει συμβεί κάθε στιγμή. Το WEKA σε κάθε ενέργεια του ενημερώνει ένα Log αρχείο με πληροφορίες για τις ενέργειες αυτές.

Παράθυρο Cluster

Όπως είναι γνωστό, μία από τις πιο συνηθισμένες εργασίες εξόρυξης γνώσης από δεδομένα είναι η συσταδοποίηση. Η συσταδοποίηση απεικονίζει τα δεδομένα σε ομάδες όπως η κατηγοριοποίηση με την διαφορά ότι οι ομάδες των δεδομένων ορίζονται από τα ίδια τα δεδομένα. Πρόκειται για μη εποπτευόμενη μάθηση και επιτυγχάνεται με τον καθορισμό της ομοιότητας, ως προς προκαθορισμένα γνωρίσματα, ανάμεσα στα δεδομένα. Τα πιο σχετικά δεδομένα ανήκουν στην ίδια ομάδα.

Το WEKA περιλαμβάνει κάποιους βασικούς αλγορίθμους συσταδοποίησης τους οποίους θα παρουσιάσουμε αναλυτικά στο κεφάλαιο 3. Στο παράθυρο Cluster του WEKA δίνεται η δυνατότητα στον χρήστη να επιλέξει κάποιον από τους αλγόριθμο συσταδοποίησης.



Εικόνα 2.3: Γραφικό περιβάλλον του WEKA - Explorer - Cluster

Όπως βλέπουμε στην εικόνα 2.3, επιλέγοντας "Clusterer" εμφανίζεται μία λίστα με όλους τους διαθέσιμους αλγορίθμους συσταδοποίησης του WEKA, από την οποία μπορούμε να επιλέξουμε τον αλγόριθμο που θα εφαρμόσουμε στα δεδομένα μας και να ρυθμίσουμε τις παραμέτρους του. Ορισμένοι αλγόριθμοι μπορούν να χρησιμοποιηθούν μόνο σε δεδομένα με ονομαστικά χαρακτηριστικά (nominal attributes), άλλοι μόνο σε δεδομένα με αριθμητικά χαρακτηριστικά (numeric attributes) και άλλοι χωρίς περιορισμό.

Με την επιλογή λειτουργίας συσταδοποίησης (Cluster Mode) γίνεται αποτίμηση των αποτελεσμάτων. Υπάρχουν τέσσερις λειτουργίες συσταδοποίησης που παρουσιάζονται παρακάτω:

- Use Training Set: Με την επιλογή αυτή αφού το WEKA ολοκληρώσει τη διαδικασία συσταδοποίησης των δεδομένων, χρησιμοποιεί το σύνολο των δεδομένων εκπαίδευσης και το κατηγοριοποιεί στις αντίστοιχες ομάδες. Στην συνέχεια υπολογίζει το ποσοστό των εγγραφών (instances) που ανήκουν στις δημιουργούμενες ομάδες (clusters).

- Supplied test set: Με την επιλογή αυτή το WEKA κάνει αποτίμηση της συσταδοποίησης σε ξεχωριστά δεδομένα ελέγχου. Η επιλογή αυτή εφαρμόζεται κυρίως σε πιθανοτικούς αλγορίθμους συσταδοποίησης όπως ο EM.

- Percentage Split: Με την επιλογή αυτή το WEKA χρησιμοποιεί το ποσοστό που επιλέγουμε, για εκπαίδευση και κάνει ομαδοποίηση στο υπόλοιπο dataset.

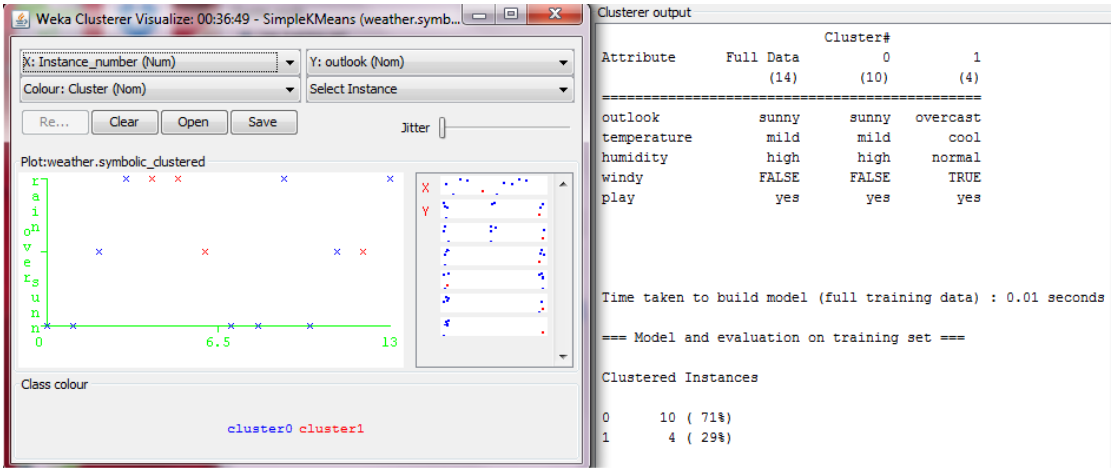
- Classes to clusters evaluation: Με την επιλογή αυτή το WEKA αρχικά αγνοεί το επιλεγμένο χαρακτηριστικό (class attribute) και ολοκληρώνει την διαδικασία της συσταδοποίησης. Στην συνέχεια κατά το στάδιο ελέγχου, χρησιμοποιώντας το χαρακτηριστικό που είχε αγνοήσει αρχικά, βρίσκει βάσει των τιμών του, σε ποια συστάδα ανήκει η κάθε τιμή, δημιουργεί τον πίνακα σύγχυσης (confusion matrix) και υπολογίζει το σφάλμα κατηγοριοποίησης.

Στα πλαίσια αυτής της εργασίας η λειτουργία συσταδοποίησης (Cluster Mode) που έχει επιλεχθεί είναι η "Use Training set".

Μια άλλη επιλογή που διαθέτει το παράθυρο Cluster είναι η επιλογή "Ignoring Attributes". Πολύ συχνά μερικά από τα χαρακτηριστικά των δεδομένων είναι απαραίτητο να εξαιρεθούν κατά

την ομαδοποίηση. Επιλέγοντας "Ignoring Attributes" εμφανίζεται παράθυρο με λίστα όλων των χαρακτηριστικών του αρχείου δεδομένων, στο οποίο είναι δυνατή η επιλογή των χαρακτηριστικών που θα εξαιρεθούν. Με την διαδικασία αυτή στην επόμενη εκκίνηση της συσταδοποίησης, τα χαρακτηριστικά αυτά θα εξαιρεθούν.

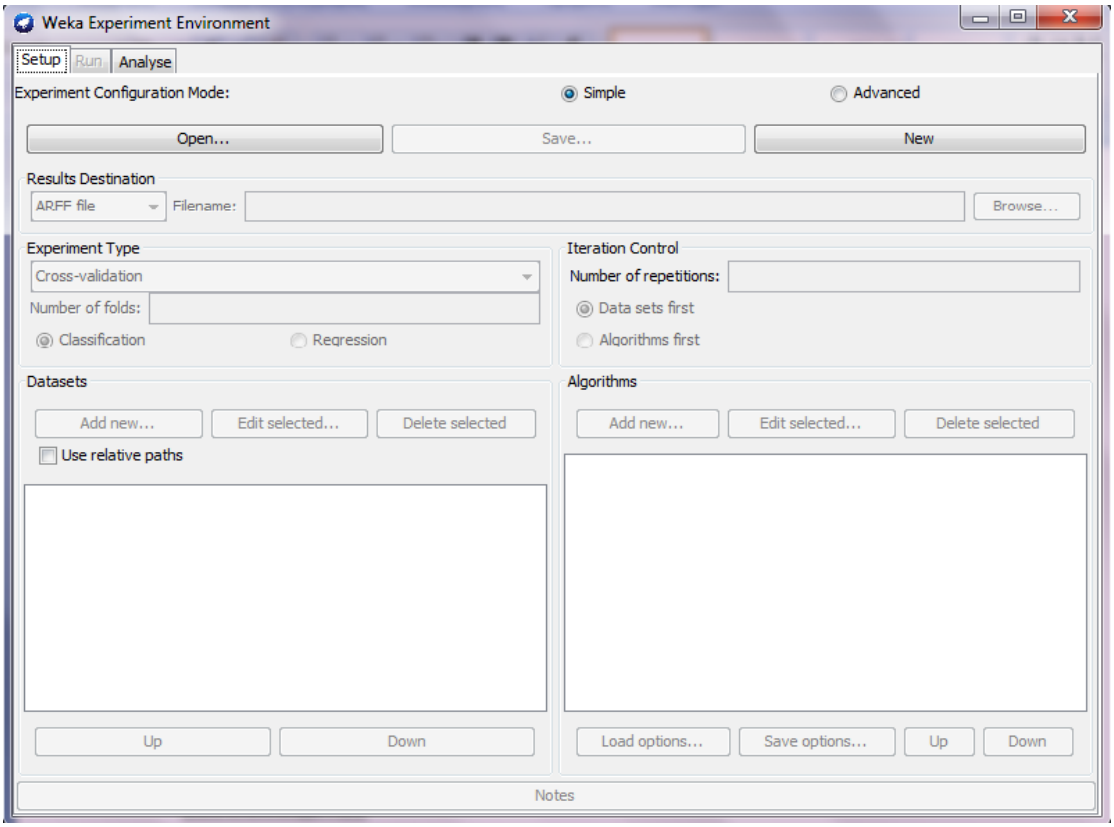
Η εκκίνηση της συσταδοποίησης πραγματοποιείται με την επιλογή "Start". Με την εμφάνιση των αποτελεσμάτων, υπάρχει δυνατότητα οπτικοποίησης τους με δεξί κλικ πάνω στον αλγόριθμο που βρίσκεται στη λίστα των αποτελεσμάτων (βλέπε εικόνα 2.4).



Εικόνα 2.4: Γραφικό περιβάλλον του WEKA - Explorer - Cluster - Οπτικοποίηση δεδομένων

Παράθυρο Experimenter

Το περιβάλλον αυτό είναι σχεδιασμένο για να απαντήσει στο βασικό πρόβλημα που αντιμετωπίζουν οι χρήστες που χρησιμοποιούν το εργαλείο αυτό, δηλαδή ποια μέθοδο και ποιες παραμέτρους πρέπει να χρησιμοποιήσουν ώστε να λάβουν το καλύτερο αποτέλεσμα. Παρά το γεγονός ότι και το περιβάλλον του Explorer μπορεί διαδραστικά να κάνει σύγκριση διαφορετικών τεχνικών μάθησης, το περιβάλλον Experimenter μπορεί να κάνει την διαδικασία πιο αυτοματοποιημένη και πιο απλή (βλέπε εικόνα 2.5).

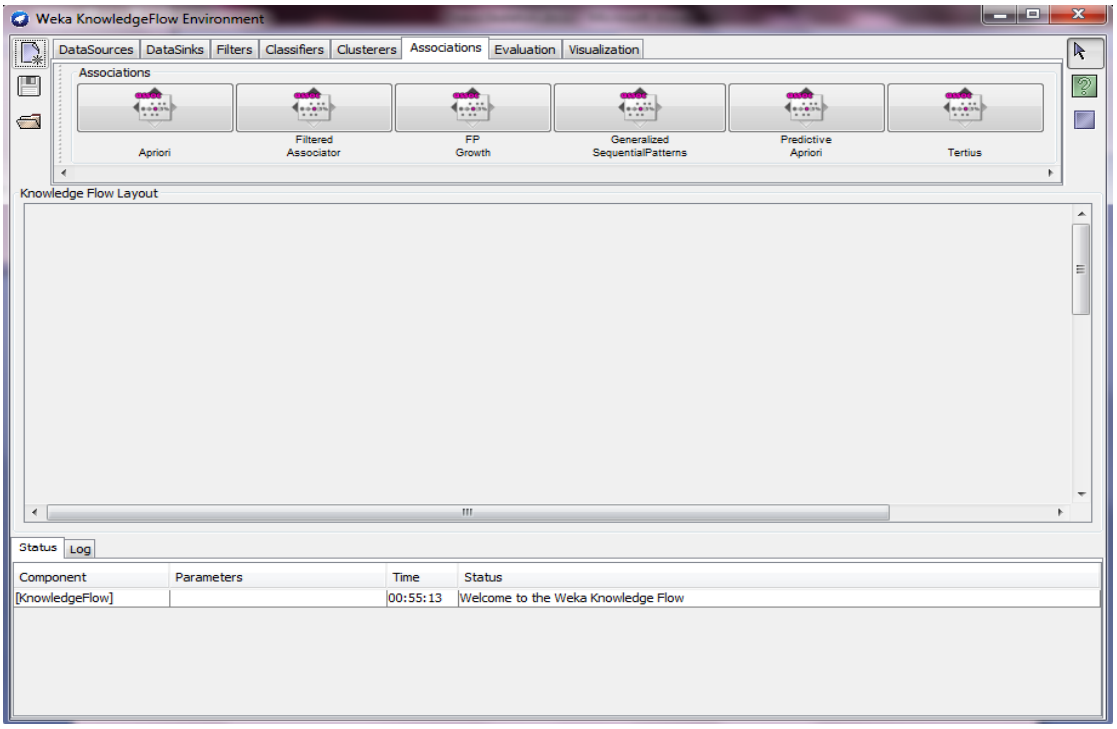


Εικόνα 2.5: Γραφικό περιβάλλον του WEKA - Experimenter

2.3.2 Knowledge Flow

Το περιβάλλον Knowledge Flow (βλέπε εικόνα 2.6) αποτελεί μια εναλλακτική πρόταση του Explorer με την διαφορά ότι προσφέρει στον χρήστη την δυνατότητα να παρακολουθήσει όλη την διαδικασία βήμα προς βήμα και όχι μόνο το αποτέλεσμα που προκύπτουν από αυτή. Το περιβάλλον αυτό αποτελεί ένα Interface ροής δεδομένων. Ο χρήστης μπορεί να επιλέξει συστατικά του WEKA από μία μπάρα εργαλείων, να τα τοποθετήσει σε ένα πλαίσιο και να τα συνδέσει μεταξύ τους προκειμένου να σχηματιστεί μια ροή γνώσης (knowledge flow) για την επεξεργασία και ανάλυση δεδομένων.

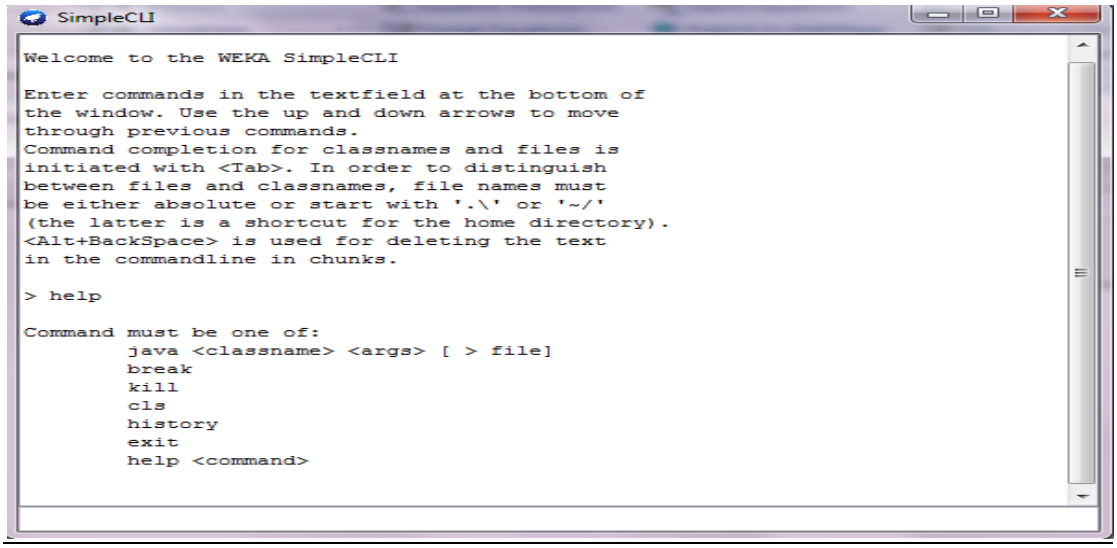
Συγκριτικά με τον Explorer, ο Knowledge Flow έχει την δυνατότητα να επεξεργαστεί δεδομένα και αυξητικά. Μια διάταξη όπου όλα τα στοιχεία της θα μπορούσαν να λειτουργήσουν αυξητικά, θα μας έδινε την δυνατότητα επεξεργασίας αρχείων οποιουδήποτε μεγέθους χωρίς να υπάρχει ο περιορισμός της κύριας μνήμης του συστήματος εφόσον δεν χρειάζεται να αποθηκεύονται τα δεδομένα εσωτερικά για να ξεκινήσει η διαδικασία.



Εικόνα 2.6: Γραφικό περιβάλλον του WEKA - Knowledge Flow

2.3.3 Simple CLI

Μέσα από το περιβάλλον του Simple CLI, ο χρήστης μπορεί να εκτελέσει τις βασικές λειτουργίες των Explorer, Knowledge Flow και Experimenter του WEKA. Όταν ο χρήστης γράφει μια εντολή (αλγόριθμο) χωρίς κάποια παράμετρο στην γραμμή εισαγωγής εντολών, στο κάτω μέρος του παραθύρου του περιβάλλοντος του, τότε εμφανίζονται όλες οι δυνατές επιλογές που συνδέονται με τον επιθυμητό αλγόριθμο. Έτσι επιλέγοντας την κατάλληλη εντολή, η αντίστοιχη λειτουργία μπορεί να επιτευχθεί.



Εικόνα 2.7: Γραφικό περιβάλλον του WEKA - Simple CLI

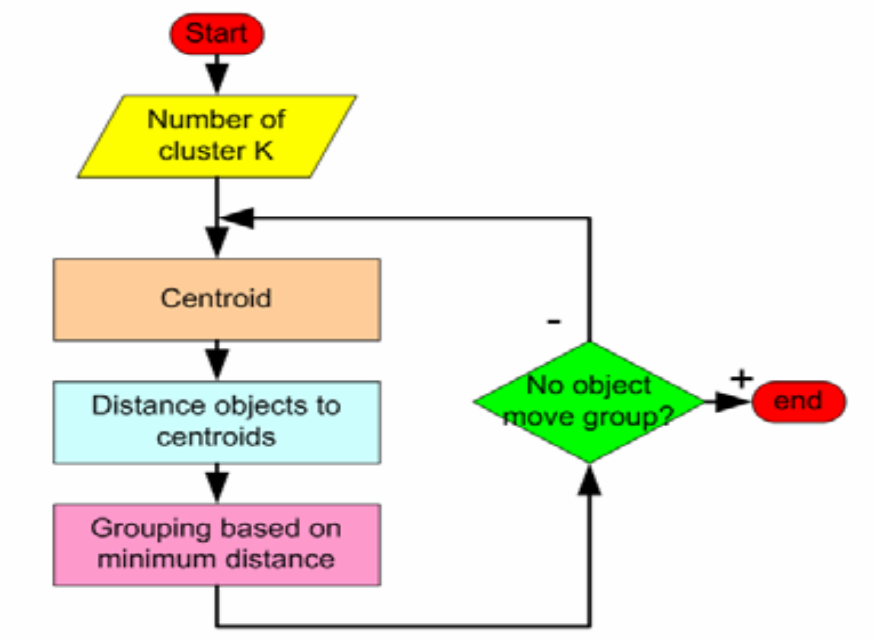
Κεφάλαιο 3 - Αλγόριθμοι Συσταδοποίησης του Λογισμικού WEKA

Όπως αναφέραμε σε προηγούμενο κεφάλαιο, μία από τις βασικές λειτουργίες του WEKA, είναι η συσταδοποίηση των δεδομένων με την χρήση κάποιων αλγορίθμων συσταδοποίησης. Παρακάτω παρουσιάζονται οι αλγόριθμοι που έχουν υλοποιηθεί στο WEKA καθώς και οι παράμετροι που δέχεται κάθε ένας από αυτούς προκειμένου να ξεκινήσει η εκτέλεσή του. Επίσης παρουσιάζονται και οι έξοδοι του κάθε αλγορίθμου.

3.1 Παρουσίαση αλγορίθμων συσταδοποίησης του λογισμικού WEKA

3.1.1 Αλγόριθμος SimpleKMeans

Ο αλγόριθμος SimpleKMeans [31] ανήκει στην κατηγορία των διαμεριστικών αλγορίθμων συσταδοποίησης και στην ευρύτερη κατηγορία των τεχνικών μάθησης χωρίς επίβλεψη. Ο αλγόριθμος αυτός είναι ιδιαίτερα δημοφιλής εξαιτίας της απλότητας της υλοποίησης του και της γραμμικής πολυπλοκότητας. Η διαδικασία της συσταδοποίησης ενός συνόλου δεδομένων με βάση τον SimpleKMeans είναι εύκολη και απλή όπως φαίνεται στο διάγραμμα ροής της παρακάτω εικόνας.

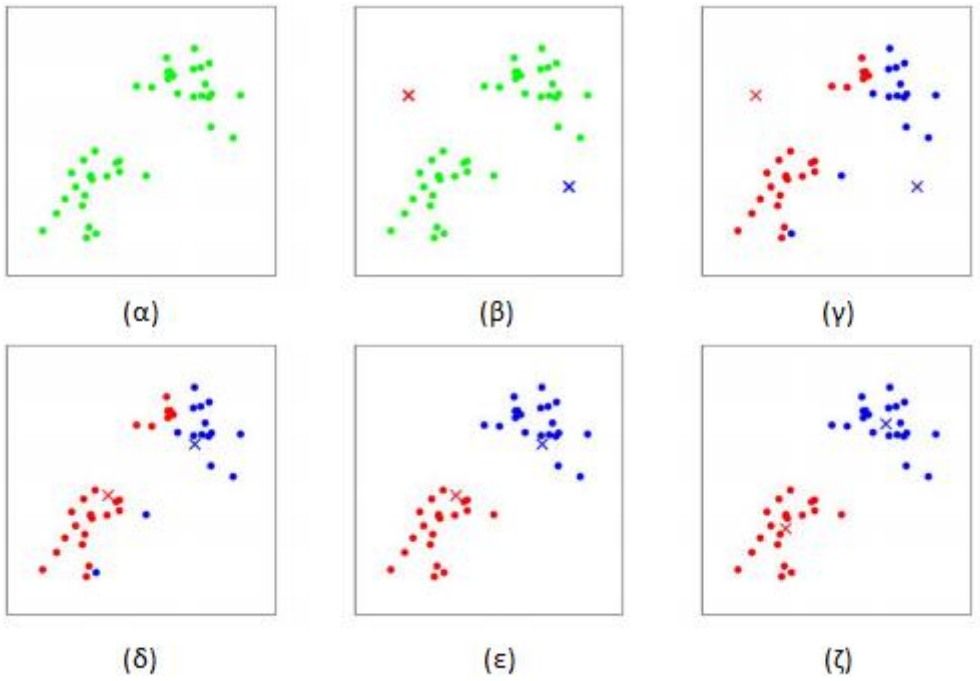


Εικόνα 3.1: Διάγραμμα ροής αλγορίθμου Simple KMeans

Η βασική ιδέα είναι να προσδιοριστούν αρχικά k κεντροειδή (centroids), ένα για κάθε συστάδα (cluster). Η επιλογή της θέσης των αρχικών κεντροειδών παίζει πολύ σημαντικό ρόλο, γιατί διαφορετικές αρχικές τους θέσεις δίνουν διαφορετικά αποτελέσματα. Καλή επιλογή της θέσης των κεντροειδών θεωρείται όταν απέχουν μεταξύ τους όσο περισσότερο γίνεται. Το επόμενο βήμα είναι η επιλογή κάθε στοιχείου από το σύνολο δεδομένων και ο υπολογισμός της ευκλείδειας απόστασης (ή άλλη μετρική απόστασης) του από τους μέσους και στη συνέχεια η τοποθέτηση του στην συστάδα από το μέσο της οποίας απέχει την μικρότερη απόσταση. Η διαδικασία αυτή ακολουθείται για όλα τα στοιχεία του συνόλου δεδομένων και έτσι ολοκληρώνεται.

Το βήμα αυτό γίνεται για όλα τα στοιχεία του συνόλου δεδομένων, το πρώτο βήμα έχει ολοκληρωθεί και μία πρώτη και «πρόχειρη» ομαδοποίηση έχει ήδη προκύψει. Στη συνέχεια,

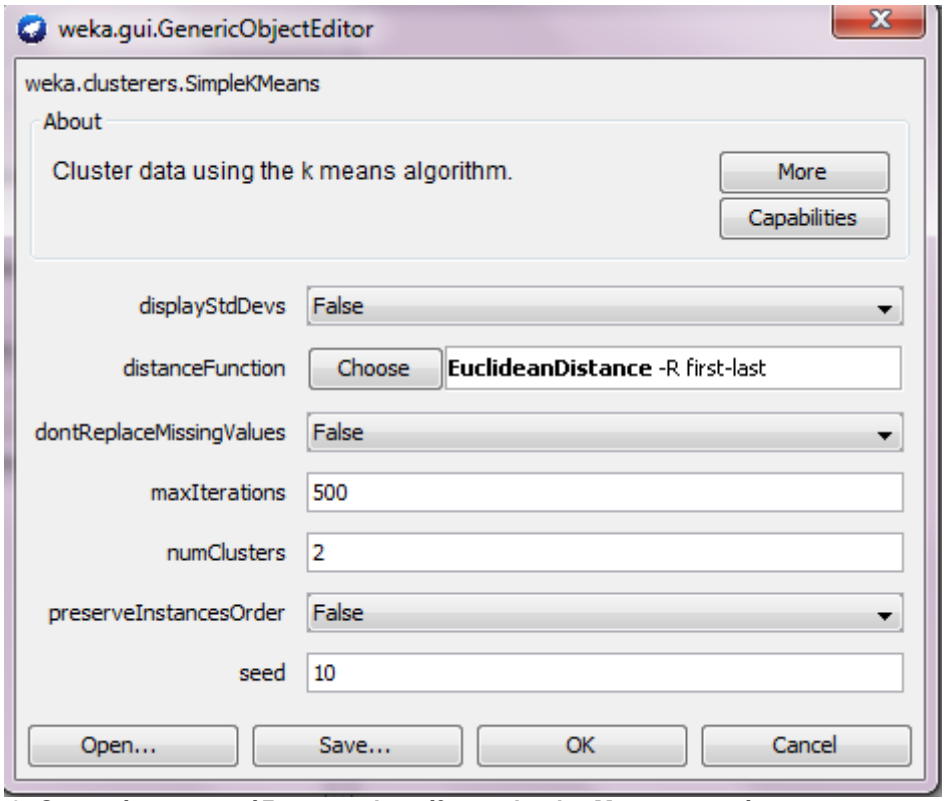
απαιτείται να υπολογιστούν ξανά k νέα κεντροειδή, τα οποία θα αποτελούν το κέντρο βάρους για κάθε μία συστάδα που προέκυψε από το προηγούμενο βήμα. Αφού λοιπόν οριστούν τα νέα k κεντροειδή, ακολουθεί και πάλι η ίδια διαδικασία ανάθεσης καθενός από τα στοιχεία του συνόλου δεδομένων στο κοντινότερο με αυτό, νέο πλέον, κεντροειδές. Έτσι, γίνεται μια επανάληψη της ίδιας διαδικασίας. Αποτέλεσμα αυτής της επανάληψης είναι ότι σε κάθε βήμα τα κεντροειδή αλλάζουν θέση (*ορίζονται νέα*) και τα στοιχεία ανατίθενται στην κατάλληλο συστάδα κάθε φορά με βάση το κοντινότερο κεντροειδές. Όταν σε κάποια επανάληψη δεν σημειωθούν αντιμεταθέσεις στοιχείων, τότε τερματίζει η εκτέλεση του αλγορίθμου. Το αποτέλεσμα που προκύπτει είναι η συσταδοποίηση του συνόλου δεδομένων σε k συστάδες.



Εικόνα 3.2: Αποτελέσματα συσταδοποίησης αλγορίθμου Simple KMeans. (α) Αρχικό αρχείο δεδομένων, (β) Τυχαία τοποθέτηση των κεντροειδών των δύο συστάδων, (γ-ζ) Παρουσιάζονται δύο επαναλήψεις του αλγορίθμου. Σε κάθε επανάληψη καταχωρούμε τα σημεία στην συστάδα με το πλησιέστερο κεντροειδές και στη συνέχεια επαναπροσδιορίζουμε την θέση του κεντροειδούς χρησιμοποιώντας αυτά τα σημεία.

Παράμετροι εισόδου του αλγορίθμου SimpleKMeans στο WEKA

Ο αλγόριθμος SimpleKMeans έχει γραμμική πολυπλοκότητα O(n * K * I * d) όπου n = αριθμός σημείων (εγγραφών) (number of points), K = αριθμός συστάδων (number of clusters), I = αριθμός επαναλήψεων (number of iterations), d = αριθμός χαρακτηριστικών (number of attribute).



Εικόνα 3.3: Οι παράμετροι εισόδου του αλγορίθμου SimpleKMeans στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο SimpleKMeans στο WEKA περιγράφονται παρακάτω.

- **DisplayStdDevs**: Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False) . Με την επιλογή της τιμής α) Αληθής , το WEKA μας παρέχει, στην έξοδό του, πληροφορία για την τυπική απόκλιση (standard deviations) των αριθμητικών χαρακτηριστικών (numeric attributes) και για το σύνολο των ονομαστικών χαρακτηριστικών (nominal attributes) ενώ αν επιλέξουμε την τιμή β) Ψευδής δεν μας επιστρέφει αυτές τις πληροφορίες.

- **DistanceFunction**: Η παράμετρος αυτή μας δίνει την δυνατότητα να επιλέξουμε τη συνάρτηση απόστασης την οποία θέλουμε να χρησιμοποιήσουμε για την ομαδοποίηση (clustering) των δεδομένων. Οι συναρτήσεις απόστασης που μας παρέχει το WEKA είναι: α)Chebyshev, β)Euclidean, γ)Manhattan , και δ)Minkowski. Αξίζει να σημειωθεί ότι ο SimpleKMeans μέχρι την έκδοση 3.7.10, χρησιμοποιεί μόνο τις συναρτήσεις Euclidean και Manhattan.

- **DontReplaceMissingValues:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False) . Με την επιλογή της τιμής α)Αληθής , όταν έχουμε περιπτώσεις δεδομένων με ελλιπείς τιμές (missing values), το WEKA μας παρέχει τη δυνατότητα αντικατάστασής τους με την μέση τιμή των διαθέσιμων δεδομένων, ενώ με την επιλογή της τιμής β) Ψευδής δεν μας δίνεται αυτή η δυνατότητα.

- **Seed:** Στην παράμετρο αυτή καθορίζεται ένας τυχαίος αριθμός αρχικών κέντρων για την εκκίνηση της διαδικασίας συσταδοποίησης του αλγορίθμου.

- **PreserveInstancesOrder:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False) . Με την επιλογή της τιμής α)Αληθής, το WEKA διατηρεί την

σειρά των δεδομένων όπως έχουν καταχωρηθεί στο αρχείο δεδομένων (dataset) ενώ με την επιλογή της τιμής β) Ψευδής το WEKA μπορεί να εναλλάσσει τη σειρά των δεδομένων κατά την ομαδοποίηση.

- **NumClusters:** Η παράμετρος αυτή καθορίζει τον αριθμό των συστάδων που θα σχηματιστούν κατά την ομαδοποίηση των δεδομένων που περιέχονται στο αρχείο εισόδου (dataset).

- **MaxIterations**: Με την παράμετρο αυτή ορίζουμε τον μέγιστο αριθμό επαναλήψεων που μπορεί να πραγματοποιήσει ο αλγόριθμος για να ολοκληρώσει την ομαδοποίηση.

- **NumExecutionSlots:** Η τιμή της παραμέτρου αυτής καθορίζει τον αριθμό των επεξεργαστών που θα χρησιμοποιήσει το WEKA για την εκτέλεση του αλγορίθμου. Η τιμή αυτή εξαρτάται από τα χαρακτηριστικά του εκάστοτε υπολογιστή στον οποίο εκτελείται η διαδικασία συσταδοποίησης. Συνήθως η τιμή της παραμέτρου ορίζεται ίση με τον αριθμό των διαθέσιμων πυρήνων (cpu/cores).

- **InitializeUsingKMeansPlusPlusMethod:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, γίνεται αρχικοποίηση των κέντρων των συστάδων βάσει της πιθανολογικής (probabilistic) μεθόδου του αλγορίθμου Farthest First ενώ με την επιλογή της τιμής β) Ψευδής χρησιμοποιείται η default διαδικασία.

- **FastDistanceCalc:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, ο αλγόριθμος χρησιμοποιεί τιμές αποκοπής (cut-off) για την βελτιστοποίηση της ταχύτητας υπολογισμού της απόστασης ενώ με την επιλογή της τιμής β) Ψευδής χρησιμοποιείται η default διαδικασία.

Τα αποτελέσματα εξόδου του αλγορίθμου SimpleKMeans όπως παρουσιάζονται στο WEKA.
Τα αποτελέσματα εξόδου του αλγορίθμου SimpleKMeans είναι τα εξής:

- Ο αριθμός επαναλήψεων, δηλαδή τις επαναλήψεις που πραγματοποίησε μέχρι να ολοκληρώσει την ομαδοποίηση των δεδομένων.

- Το Sum of Squared Error, δηλαδή το άθροισμα των αποστάσεων του κάθε στοιχείου κάθε ομάδας (cluster) ως προς τα αντίστοιχα κέντρα (υψωμένα στο τετράγωνο).

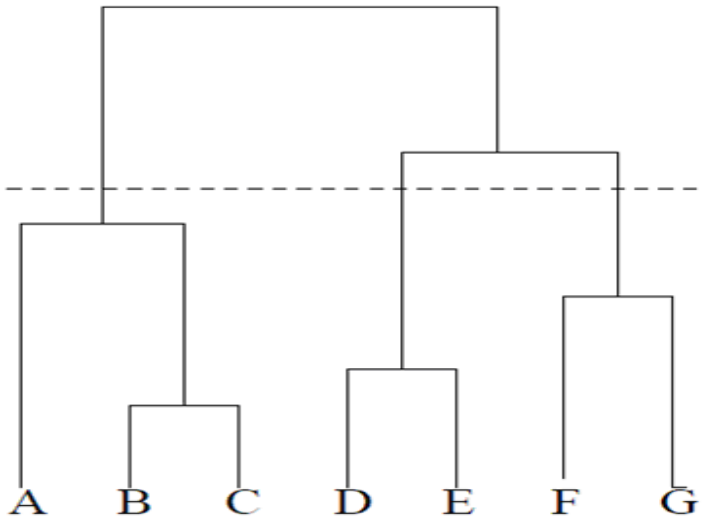
- Η μέση τιμή του κάθε χαρακτηριστικού για κάθε συστάδα.

- Το σύνολο και το ποσοστό των εγγραφών για κάθε συστάδα.

- Ο χρόνος ολοκλήρωσης της συσταδοποίησης.

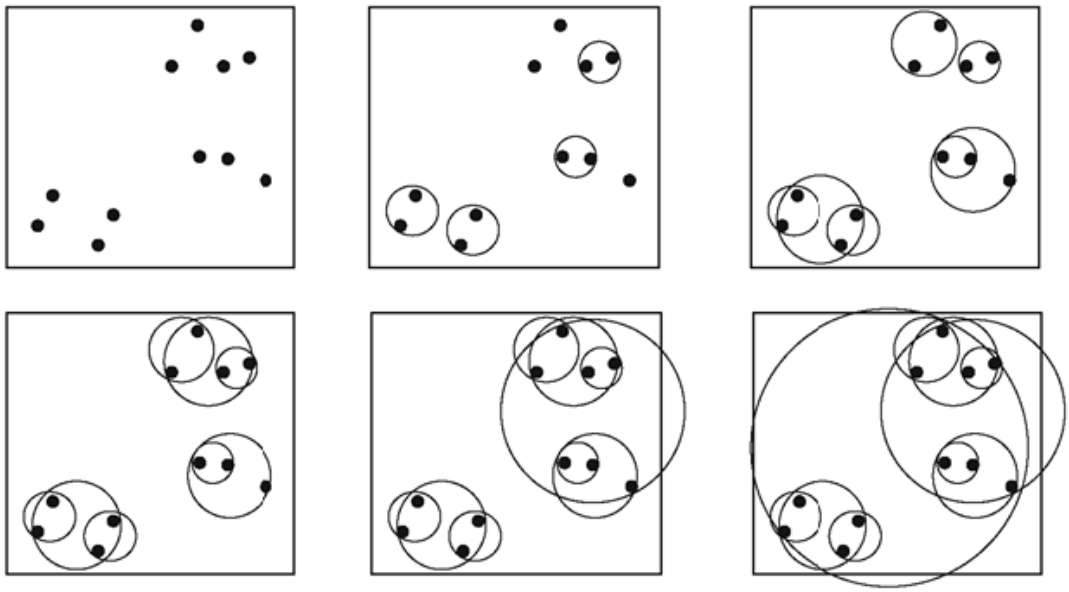
3.1.2 Αλγόριθμος Hierarchical Clusterer

Στην ιεραρχική ομαδοποίηση τα αποτελέσματα συσταδοποίησης δίνονται με μορφή δενδρογράμματος. Στα δενδρογράμματα αυτά, επιλέγεται ένα επίπεδο στο οποίο θα "κοπούν". Το σημείο στο οποίο θα κόψουμε κάποιο δενδρόγραμμα (βλέπε εικόνα 3.4), δείχνει τον αριθμό των συστάδων που θα προκύψουν καθώς και τα σημεία που περιέχει η κάθε συστάδα. Οι ιεραρχικές τεχνικές χωρίζονται στις Συσσωρευτικές και Διαιρετικές.



Εικόνα 3.4: Παράδειγμα δενδρογράμματος 7 στοιχείων τα οποία συσταδοποιούνται σε τρεις συστάδες (βλέπε διακεκομμένη γραμμή).

Οι Συσσωρευτικές (Agglomerative) τεχνικές αρχικά θεωρούν ότι κάθε σημείο είναι από μόνο του μια ξεχωριστή συστάδα και προχωρούν σε συγχωνεύσεις αυτών με βάση μετρικές αποστάσεων όπως Ευκλείδεια, Mahalanobis κ.α., μέχρι όλα τα σημεία να τοποθετηθούν σε μία συστάδα. Αυτή η τεχνική ονομάζεται από κάτω προς τα πάνω (bottom up, βλέπε εικόνα 3.4).



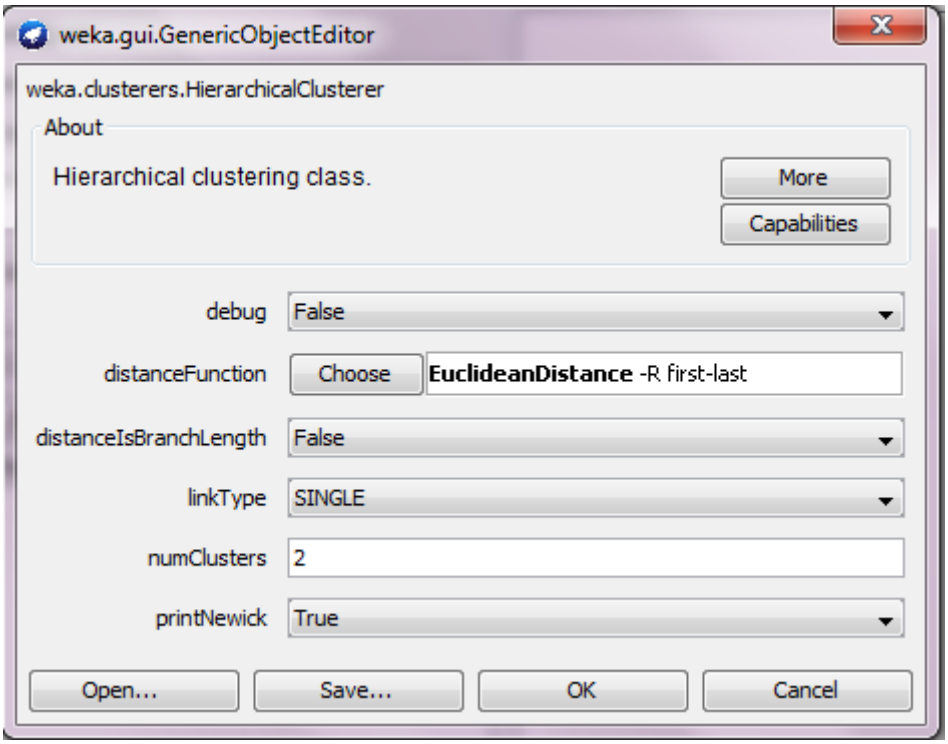
Εικόνα 3.5: Παράδειγμα εκτέλεσης συσσωρευτικού - ιεραρχικού αλγορίθμου. Τα κοντινότερα σημεία συσσωρεύονται διαδοχικά σε συστάδες και στη συνέχεια γίνεται συγχώνευση των κοντινότερων συστάδων μέχρι να καταλήξουμε σε μία υπερσυστάδα.

Οι Διαιρετικές (Divisive) τεχνικές λειτουργούν αντίστροφα. Αρχικά θεωρούν ότι όλα τα σημεία ανήκουν σε μία ομάδα και στη συνέχεια την διαιρούν έως ότου κάθε συστάδα να περιέχει ένα μόνο σημείο. Η τεχνική αυτή ονομάζεται από πάνω προς τα κάτω (*top-down*).

Αν και η μέθοδος των ιεραρχικών αλγόριθμων είναι πολύ χρήσιμη σε διάφορες εφαρμογές όπου απαιτείται συσταδοποίηση και ο αριθμός των συστάδων δεν είναι γνωστός εκ των προτέρων, παρουσιάζουν το σημαντικό μειονέκτημα της αυξημένης πολυπλοκότητας (μεγαλύτερη από $O(n^2)$) με αποτέλεσμα να μην μπορούν να εφαρμοστούν άμεσα σε αρχεία που περιέχουν μεγάλο αριθμό εγγραφών και χαρακτηριστικών.

Παράμετροι εισόδου του αλγορίθμου Hierarchical στο WEKA

Ο Hierarchical αλγόριθμος έχει υψηλή πολυπλοκότητα $O(n^2)$, όπου n = αριθμός σημείων.



Εικόνα 3.6: Οι παράμετροι εισόδου του αλγορίθμου Hierarchical στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο Hierarchical στο WEKA περιγράφονται παρακάτω.

- **debug:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, έχουμε την δυνατότητα να λάβουμε επιπλέον πληροφορίες στην εκτέλεση του αλγορίθμου.
- **linkType:** Η παράμετρος αυτή καθορίζει την μέθοδο που θα χρησιμοποιήσει ο αλγόριθμος για τον υπολογισμό της απόστασης μεταξύ δύο συστάδων. Οι διαθέσιμες μέθοδοι είναι οι παρακάτω:
    - SINGLE: Βρίσκει την μικρότερη απόσταση μεταξύ ενός σημείου της μιας ομάδας και ενός σημείου της άλλης ομάδας.
    - COMPLETE: Βρίσκει την μεγαλύτερη απόσταση μεταξύ ενός σημείου της μιας ομάδας και ενός σημείου της άλλης ομάδας.
    - ADJCOMLPETE: Βρίσκει την μεγαλύτερη απόσταση μεταξύ ενός σημείου της μιας ομάδας και ενός σημείου της άλλης ομάδας, όπως η μέθοδος Complete, αλλά χρησιμοποιεί την μεγαλύτερη απόσταση όλων των συστάδων.
    - AVERAGE: Βρίσκει την μέση τιμή της απόστασης μεταξύ των σημείων δύο συστάδων.
    - MEAN: Υπολογίζει την μέση απόσταση των σημείων μιας συγχωνευμένης κλάσης.

- CENTROID: Βρίσκει την απόσταση των κέντρων των συστάδων.
- WARD: Βρίσκει την αλλαγή των αποστάσεων που προκαλείται από την συγχώνευση των συστάδων.
- NEIGHBOR_JOINING: Χρησιμοποιεί τον αλγόριθμο neighbor joining.

- **distanceFunction:** Η παράμετρος αυτή καθορίζει την συνάρτηση απόστασης με την οποία θα γίνει ο υπολογισμός της απόστασης μεταξύ δύο σημείων ή μεταξύ ενός σημείου με το κέντρο της ομάδας. Εξαρτάται από την επιλογή της παραμέτρου linkType.
- **distanceIsBranchLength:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, η απόσταση μεταξύ δύο συστάδων ερμηνεύεται ως το μήκος του κλαδιού του δενδροδιαγράμματος ενώ με την επιλογή β)Ψευδής η απόσταση μεταξύ των συστάδων ερμηνεύεται ως το ύψος των κόμβων που συνδέουν τις ομάδες. Στην περίπτωση όπου η παράμετρος Link type έχει την τιμή Neighbor Joining, η τιμή πρέπει να είναι Ψευδής, διαφορετικά δεν είναι δυνατή η απεικόνιση του δενδροδιαγράμματος.
- **numClusters:** Η παράμετρος αυτή καθορίζει τον αριθμό των συστάδων που θα δημιουργήσει ο αλγόριθμος.
- **printNewick:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, η απεικόνιση των συστάδων γίνεται σε Newick μορφή (τρόπος αναπαράστασης γραφημάτων-δέντρων με παρενθέσεις και κόμματα). Δεν ενδείκνυται η χρήση αυτής της μορφής σε μεγάλα αρχεία δεδομένων.

Τα αποτελέσματα εξόδου του αλγορίθμου Hierarchical όπως παρουσιάζονται στο WEKA.

Τα αποτελέσματα εξόδου του αλγορίθμου Hierarchical είναι τα εξής:

- Ο χρόνος ολοκλήρωσης της συσταδοποίησης.
- Το σύνολο και το ποσοστό των εγγραφών (instances) για κάθε συστάδα.
- Όταν στη παράμετρο printNewick του αλγορίθμου είναι επιλεγμένη η τιμή Αληθής τότε λαμβάνουμε σαν έξοδο την απεικόνιση των συστάδων σε Newick μορφή (πχ. Cluster 0 (0.0:2,49418,(0.0:2,26403,0.0:2,26403):0,23015)).

3.1.3 Αλγόριθμος CobWeb

Ο αλγόριθμος Cobweb [32] είναι ένας εννοιολογικός αλγόριθμος συσταδοποίησης ο οποίος αναπτύχθηκε στις αρχές του 1980 για την κατηγοριοποίηση αντικειμένων ενός συνόλου δεδομένων. Είναι ένας αυξητικός αλγόριθμος ο οποίος δεν απαιτεί εκ των προτέρων γνώση του αριθμού των ομάδων.

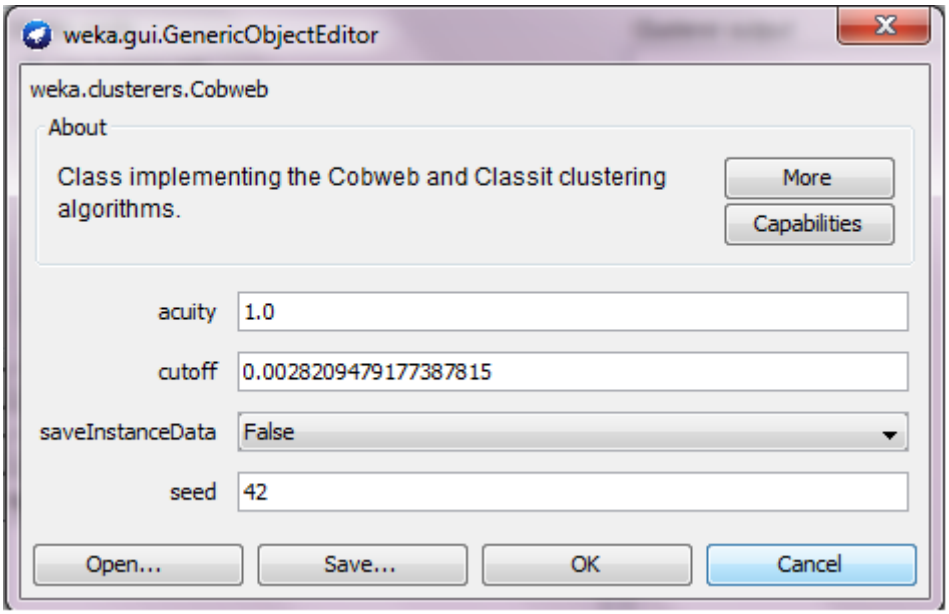
Βασίζεται στον σχεδιασμό ενός δενδρογράμματος συσταδοποίησης στο οποίο κάθε συστάδα χαρακτηρίζεται από μία πιθανολογική (probabilistic) περιγραφή. Η περιγραφή αυτή γίνεται χρησιμοποιώντας μία συνάρτηση (Category Utility function - CU) η οποία μετράει την ποιότητα συσταδοποίησης, δηλαδή πόσο καλά ταιριάζει ένα αντικείμενο σε μία ομάδα.

Το δενδροδιάγραμμα που κατασκευάζει ο Cobweb γίνεται αυξητικά, εισάγοντας τα αντικείμενα του dataset στο δένδρο, βήμα προς βήμα. Όταν εισάγεται ένα αντικείμενο, ο αλγόριθμος αναδιαμορφώνει την δομή του δένδρου από πάνω προς τα κάτω, ξεκινώντας από τον αρχικό κόμβο (root node). Για την αναδιαμόρφωση του δένδρου, ο αλγόριθμος χρησιμοποιεί τέσσερις πιθανές λειτουργίες (εισαγωγή, δημιουργία, συγχώνευση και διαχώριση). Η λειτουργία που θα επιλεγεί για την αναδιαμόρφωση του δένδρου βασίζεται στις αντίστοιχες τιμές της συνάρτησης CU που

συγκεντρώνει η κάθε μία. Η διαδικασία εισαγωγής σημαίνει ότι ένα νέο αντικείμενο εισάγεται σε έναν από τους ήδη υπάρχοντες κόμβους. Για να γίνει η εισαγωγή, ο Cobweb υπολογίζει την τιμή CU του νεοεισερχόμενου αντικειμένου για κάθε έναν από τους κόμβους του δένδρου και τον τοποθετεί σε αυτόν όπου η CU έχει την μεγαλύτερη τιμή. Με παρόμοιες τεχνικές ο αλγόριθμος Cobweb, υπολογίζει τις τιμές της συνάρτησης CU και αντιστοίχως αποφασίζει για τις λειτουργίες της εισαγωγής, δημιουργίας ενός κόμβου, της συγχώνευσης δύο κόμβων (αυτών με τις μεγαλύτερες τιμές της συνάρτησης CU) και για τη διαχώριση.

Παράμετροι εισόδου του αλγορίθμου Cobweb στο WEKA

Ο αλγόριθμος Cobweb έχει υψηλή πολυπλοκότητα $O(d\ n\ b^2\ \log\ k)$ όπου n = αριθμός σημείων (number of points), K = αριθμός συστάδων (number of clusters), I = αριθμός επαναλήψεων (number of iterations), d = αριθμός χαρακτηριστικών (number of attribute), b = παράγοντας διακλάδωσης (branching factor).

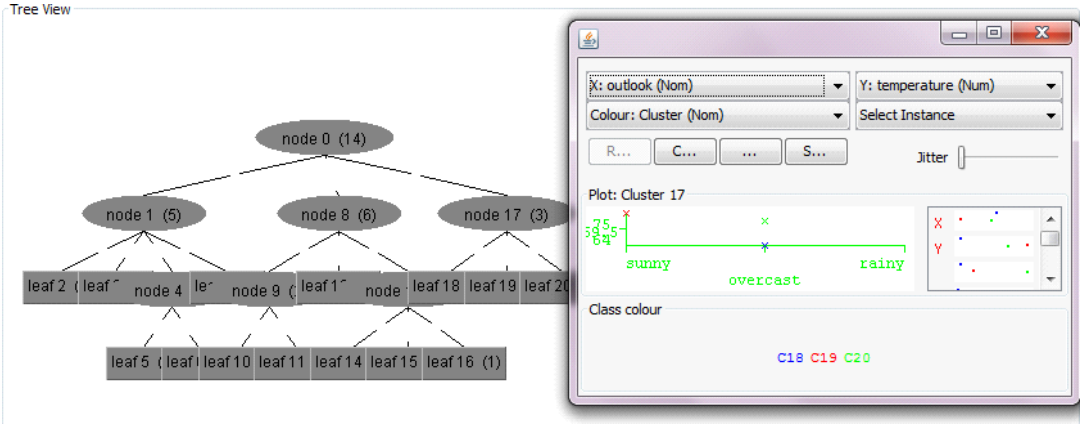


Εικόνα 3.7: Οι παράμετροι εισόδου του αλγορίθμου CobWeb στο σύστημα WEKA

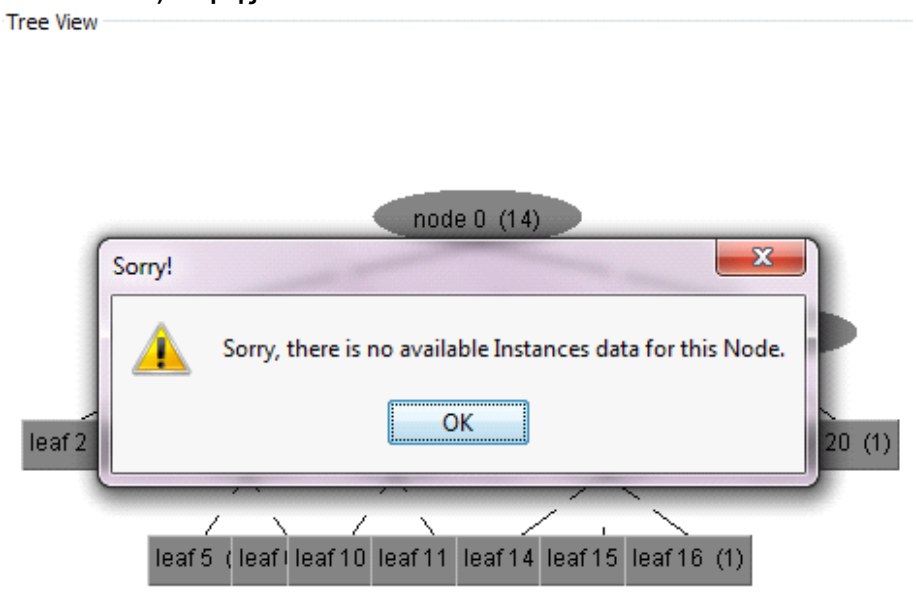
Οι παράμετροι εισόδου που δέχεται ο Cobweb στο WEKA περιγράφονται παρακάτω:

- **Seed:** Στην παράμετρο αυτή καθορίζεται ένας τυχαίος αριθμός αρχικών κέντρων για την εκκίνηση της διαδικασίας συσταδοποίησης του αλγορίθμου.
- **SaveInstanceData:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, έχουμε την δυνατότητα να αποθηκεύσουμε τις πληροφορίες του παραγόμενου δενδροδιαγράμματος ώστε να μπορούμε ανά πάσα στιγμή να απεικονίσουμε τα παραγόμενα αποτελέσματα του αλγορίθμου (Με δεξί κλικ στην λίστα αποτελεσμάτων, επιλέγουμε Visualize Tree). Όταν η επιλογή της τιμής είναι β) Ψευδής τότε δεν έχουμε διαθέσιμες πληροφορίες κατά την απεικόνιση του αποτελέσματος του αλγορίθμου.

Βλέπε σχήματα:



Εικόνα 3.8: α) Αληθής



Εικόνα 3.9: β) Ψευδής

- **cutoff:** Η παράμετρος αυτή καθορίζει το κατώφλι (threshold) της συνάρτησης CU (Category Utility) βάσει της οποίας γίνεται αποκοπή (cutoff) του δενδροδιαγράμματος. Όσο μεγαλύτερη είναι η τιμή αποκοπής, ο CobWeb δημιουργεί λιγότερες ομάδες και επομένως υπάρχει μεγαλύτερη ανομοιομορφία μεταξύ των εγγραφών που εντάσσονται στην ίδια ομάδα. Αν η τιμή παραμέτρου αποκοπής πάρει την τιμή 0 τότε ο αλγόριθμος αντιστοιχίζει κάθε εγγραφή σε μία ομάδα. Η παράμετρος αυτή παίρνει τιμές στο εύρος [0, 1] και η αρχική τιμή της είναι 0.002.

- **acuity :** Η παράμετρος αυτή καθορίζει την ελάχιστη τιμή της τυπικής απόκλισης που απαιτούμε να έχουν μεταξύ τους οι εγγραφές του συνόλου δεδομένων μας. Αξίζει σημειωθεί ότι η επιλογή αυτή μας δίνεται μόνο όταν επιλέξουμε χαρακτηριστικά τα οποία περιγράφονται από αριθμητικές τιμές. Τα δεδομένα με τυπική απόκλιση μικρότερη της δοθείσας τιμής της παραμέτρου, δεν λαμβάνουν μέρος στην ομαδοποίηση. Η αρχική τιμή αυτής της παραμέτρου είναι 0.1.

Τα αποτελέσματα εξόδου του αλγορίθμου Cobweb όπως παρουσιάζονται στο WEKA.

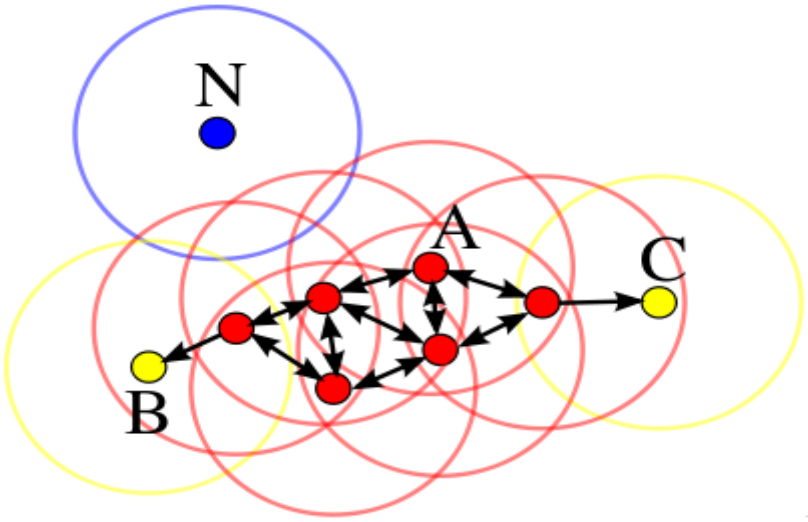
Τα αποτελέσματα εξόδου του αλγορίθμου Cobweb είναι τα εξής:

- Οι συγχωνεύσεις, διαιρέσεις και ο αριθμός των συστάδων.
- Το δενδροδιάγραμμα.
- Ο χρόνος ολοκλήρωσης της συσταδοποίησης.
- Το σύνολο και το ποσοστό των εγγραφών για κάθε συστάδα.

3.1.4 Αλγόριθμος DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Ο DBSCAN [33] είναι ένα αλγόριθμος ο οποίος βασίζεται στην χωρική πυκνότητα των δεδομένων. Ο αλγόριθμος δημιουργεί συστάδες σε περιοχές με υψηλή πυκνότητα και έχει τη δυνατότητα να ανακαλύπτει συστάδες ακανόνιστου σχήματος σε χωρικές βάσεις δεδομένων. Απαιτεί την εισαγωγή δύο παραμέτρων, της ελάχιστης ακτίνας (*Eps*) και του ελάχιστου αριθμού σημείων (*MinPts*). Η γειτονιά εντός μίας ακτίνας Eps ενός δεδομένου αντικειμένου καλείται Eps – γειτονιά του αντικειμένου και ένα αντικείμενο με αριθμό αντικειμένων τουλάχιστον ίσο με MinPts εντός της Eps – γειτονιάς του καλείται πυρήνας. Προκειμένου να ανακαλύψει ομάδες στα δεδομένα, ο DBSCAN, ελέγχει την Eps – γειτονιά κάθε σημείου στην βάση δεδομένων. Εάν η Eps – γειτονιά ενός σημείου p περιέχει περισσότερα από MinPts, δημιουργείται μία νέα συστάδα με πυρήνα το p. Τα σημεία (*πυρήνες*) εντός των νεοπροστιθέμενων μελών θα υποστούν την ίδια διαδικασία με το p, προκειμένου να αυξήσουν το μέγεθος της ομάδας. Έτσι όταν σταματήσουν να εντοπίζονται πυρήνες με αυτήν την διαδικασία, ένα άλλο νέο σημείο (*πυρήνας*) θα ληφθεί από την βάση δεδομένων προκειμένου να εξελίξει μία νέα ομάδα.

Αξίζει να σημειωθεί ότι κατά την διάρκεια της αυξητικής διαδικασίας, ένα σημείο (*πυρήνας*) το οποίο ήδη ανήκει σε μία άλλη συστάδα μπορεί να χρησιμοποιηθεί και από άλλη συστάδα, κάτι το οποίο θα οδηγήσει στην συνένωση των δύο συστάδων. Ο αλγόριθμος τερματίζει όταν κανένα νέο σημείο δεν μπορεί να προστεθεί στην συστάδα.



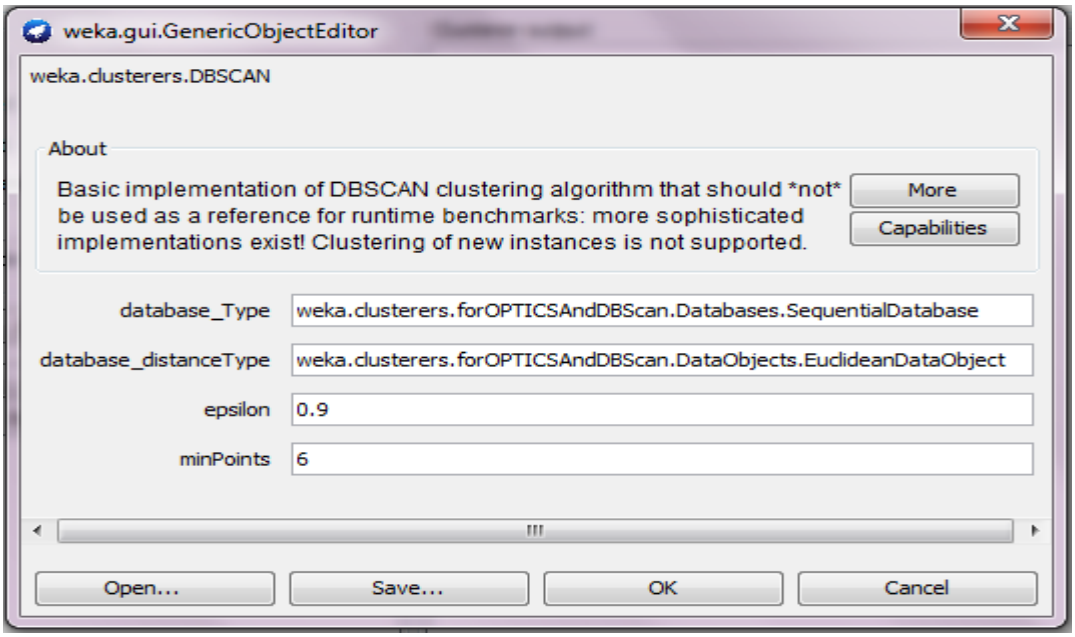
Εικόνα 3.10: Παράδειγμα εκτέλεσης του αλγορίθμου DBSCAN

Στην εικόνα 3.10 δίνεται ένα παράδειγμα του αλγορίθμου DBSCAN για τιμή MinPts = 3. Το σημείο A και τα υπόλοιπα κόκκινα σημεία αποτελούν σημεία (πυρήνες) διότι έχουν τουλάχιστον 3 γειτονικά σημεία τα οποία περιέχονται μέσα σε κύκλους με ακτίνα ίση με ε. Επειδή τα σημεία αυτά επικοινωνούν μεταξύ του (ανήκουν σε κοινούς κύκλους), θεωρούμε ότι ανήκουν στην ίδια συστάδα. Τα σημεία B και C παρόλα αυτά δεν θεωρούνται ως σημεία (πυρήνες), διότι οι κύκλοι τους δεν

έχουν τουλάχιστον 3 σημεία και επομένως δεν ανήκουν στη συστάδα. Τέλος το σημείο N θεωρείται ως θόρυβος (noise point) μιας και δεν είναι ούτε σημείο πυρήνας αλλά ούτε αποτελεί σημείο που συνδέεται με τα σημεία της συστάδας.

Παράμετροι εισόδου του αλγορίθμου DBSCAN στο WEKA

Ο DBSCAN αλγόριθμος έχει υψηλή πολυπλοκότητα $O(n^2)$ όπου n = αριθμός σημείων (number of points).



Εικόνα 3.10: Οι παράμετροι του αλγορίθμου DBSCAN στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο DBSCAN στο WEKA περιγράφονται παρακάτω:

- **database_Type**: Η παράμετρος αυτή καθορίζει τον τύπο της βάσης που θα χρησιμοποιηθεί κατά την εκτέλεση του αλγορίθμου για την αποθήκευση του αρχικού αρχείου δεδομένων.

- **database_distanceType:** Η τιμή αυτής της παραμέτρου καθορίζει την μετρική αποστάσεων που θα χρησιμοποιηθεί από τον αλγόριθμο όπως για παράδειγμα η Ευκλείδεια.

- **epsilon**: Με την παράμετρο αυτή ορίζουμε την ελάχιστη ακτίνα που καθορίσει την περιοχή των σημείων τα οποία θα ελεγχθούν κατά την εκκίνηση της διαδικασίας συσταδοποίησης.

- **minPoints**: Η τιμή αυτής της παραμέτρου ορίζει τον ελάχιστο αριθμό σημείων που απαιτείται να περιέχονται μέσα την κυκλική περιοχή ενός σημείου (πυρήνα).

Τα αποτελέσματα εξόδου του αλγορίθμου DBSCAN όπως παρουσιάζονται στο WEKA.

Τα αποτελέσματα εξόδου του αλγορίθμου DBSCAN είναι τα εξής:

- Ο χρόνος ολοκλήρωσης της συσταδοποίησης.

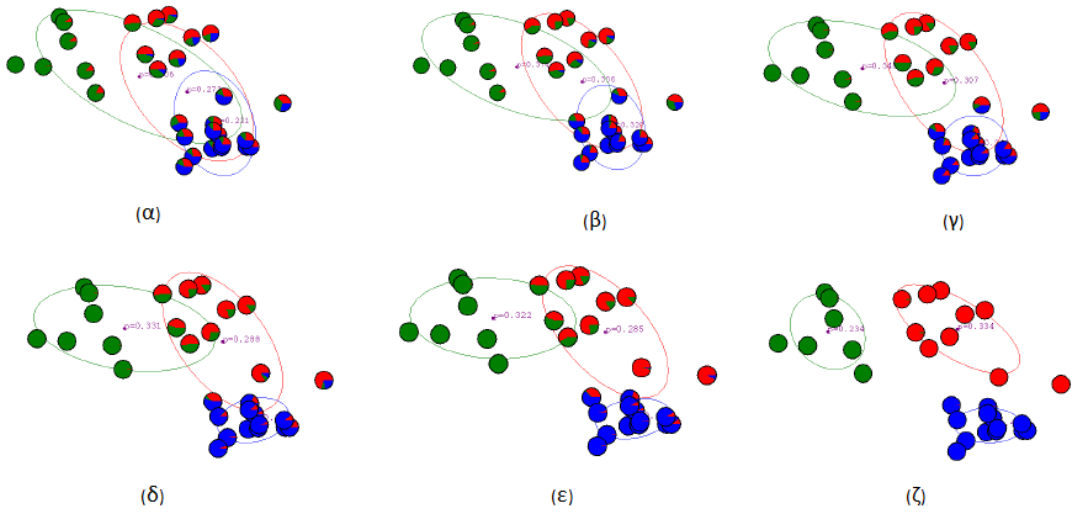
- Η συστάδα που ανήκει κάθε εγγραφή του συνόλου εγγραφών του αρχείου.

- Το σύνολο και το ποσοστό των εγγραφών για κάθε συστάδα.

3.1.5 Αλγόριθμος EM (Expectation Maximization)

Ο αλγόριθμος EM [34] είναι ένας πολύ βασικός αλγόριθμος ο οποίος χρησιμοποιείται ευρέως σε τεχνικές data mining, και βρίσκει εφαρμογή σε προβλήματα όπου έχουμε ελλιπή δεδομένα (missing data). Ο αλγόριθμος αυτός είναι ένας επαναληπτικός αλγόριθμος ο οποίος εκτελεί δύο βασικά βήματα, τα οποία εναλλάσσονται μεταξύ τους. Το πρώτο βήμα ονομάζεται βήμα ευρέσεως προσδοκώμενης τιμής (expectation step). Σκοπός του είναι να υπολογίσει την αναμενόμενη τιμή μιας πιθανότητας που εκτιμάται χρησιμοποιώντας τις τρέχουσες εκτιμήσεις των παραμέτρων ενός στατιστικού μοντέλου (π.χ. μιας κανονικής κατανομής). Το δεύτερο βήμα είναι το βήμα μεγιστοποίησης (maximization step) το οποίο υπολογίζει τις παραμέτρους του στατιστικού μοντέλου, μεγιστοποιώντας την αναμενόμενη τιμή πιθανότητας που βρέθηκε στο προηγούμενο βήμα (E step). Οι εκτιμημένες αυτές παράμετροι στη συνέχεια χρησιμοποιούνται ως οι νέες τιμές των παραμέτρων του στατιστικού μοντέλου που χρησιμοποιείται στο πρώτο βήμα (E step).

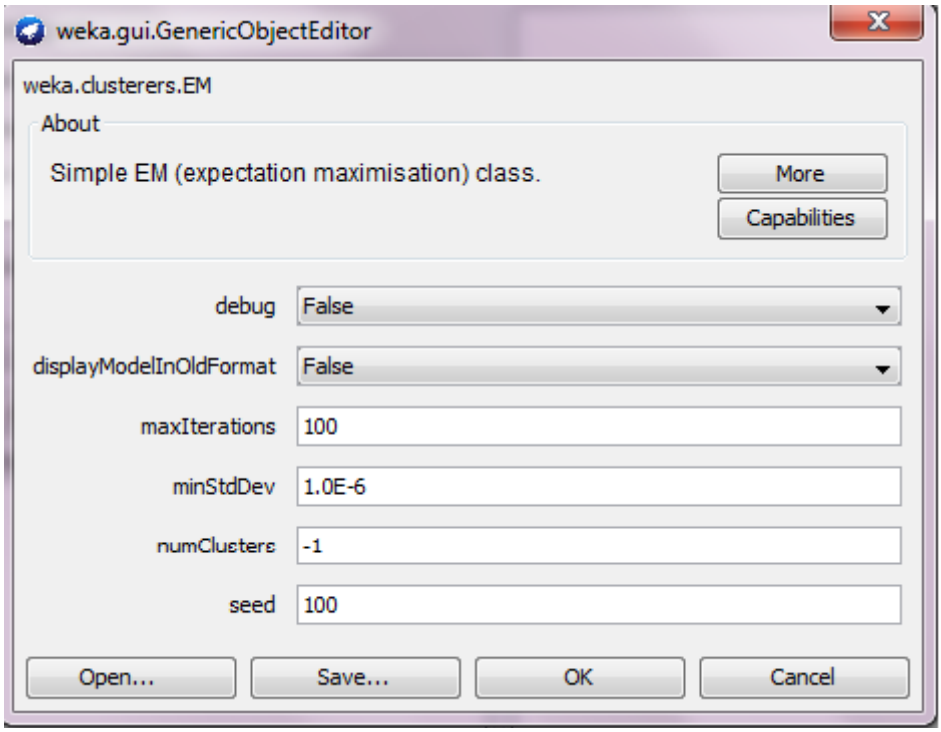
Στην εικόνα 3.11 βλέπουμε τις πρώτες 6 επαναλήψεις του αλγορίθμου EM. Στο σχήμα (α) φαίνεται η αρχικοποίηση των μοντέλων που είναι και η αρχική γνώση μας για το που ανήκουν τα δεδομένα. Όπως παρατηρούμε οι κουκίδες είναι χρωματισμένες με βάση την πιθανότητα τους να ανήκουν σε κάποια από τις 3 ομάδες. Καθώς εξελίσσεται ο αλγόριθμος και γίνεται ο επανυπολογισμός των παραμέτρων των μοντέλων τα ποσοστά κάθε χρώματος κάθε κουκίδας αλλά και τα κέντρα και το μέγεθος των ελλείψεων της κάθε ομάδας αλλάζουν (βλέπε σχήματα β,γ,δ,ε,ζ). Στο σχήμα (ζ) παρατηρούμε ότι η διασπορά του κάθε μοντέλου έχει μειωθεί και ότι ο διαχωρισμός των ομάδων γίνεται ξεκάθαρος. Ο αλγόριθμος τερματίζει όταν οι μεταβολές στις τιμές των παραμέτρων των μοντέλων γίνουν αμελητέες από βήμα σε βήμα.



Εικόνα 3.11: Παράδειγμα αλγορίθμου EM.

Παράμετροι εισόδου του αλγορίθμου EM στο WEKA

Ο αλγόριθμος EM έχει υψηλή πολυπλοκότητα *O(n * K * I * d)* όπου n = αριθμός σημείων, K = αριθμός συστάδων, I = αριθμός επαναλήψεων (number of iterations), d = αριθμός χαρακτηριστικών.



Εικόνα 3.12: Οι παράμετροι του αλγορίθμου EM στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο EM στο WEKA περιγράφονται παρακάτω.

- **debug:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, έχουμε την δυνατότητα να λάβουμε επιπλέον πληροφορίες στην έξοδο του αλγορίθμου ενώ με την επιλογή της τιμής β) Ψευδής λαμβάνουμε τις βασικές πληροφορίες εξόδου, όπως περιγράφονται παρακάτω.

- **minLogLikelihoodImprovementIterating:** Η τιμή της παραμέτρου αυτής καθορίζει την ελάχιστη βελτίωση της λογαριθμικής πιθανοφάνειας (log likelihood) που απαιτείται για την εκτέλεση επιπλέον επανάληψης του αλγορίθμου.

- **minStdDev:** Η παράμετρος αυτή καθορίζει την ελάχιστη επιτρεπτή τιμή της τυπικής απόκλισης μεταξύ των εγγραφών που θα χρησιμοποιηθούν στην ομαδοποίηση. Αξίζει να σημειωθεί ότι για να χρησιμοποιηθεί αυτή η παράμετρος θα πρέπει να έχουμε εγγραφές που περιγράφονται από αριθμητικά χαρακτηριστικά**.**

- **seed:** Στην παράμετρο αυτή καθορίζεται ένας τυχαίος αριθμός αρχικών κέντρων για την εκκίνηση της διαδικασίας συσταδοποίησης του αλγορίθμου.

- **minLogLikelihoodImprovementCV:** Η τιμή της παραμέτρου καθορίζει την ελάχιστη μεταβολή της λογαριθμικής πιθανοφάνειας (log likelihood) που απαιτείται κατά το Cross Validation, προκειμένου να εξεταστεί το ενδεχόμενο αύξησης του αριθμού των συστάδων (clusters) για την καλύτερη εύρεση αυτών. (Κατά την διαδικασία Cross Validation, ο EM καθορίζει τον αριθμό των συστάδων που θα δημιουργηθούν.)

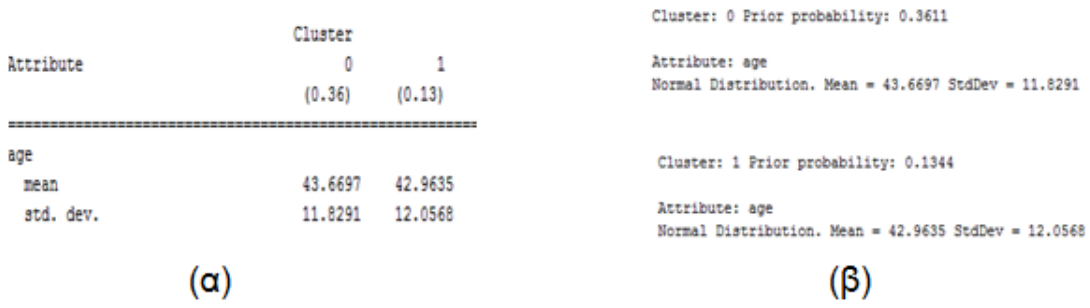
- **numClusters:** Η παράμετρος αυτή καθορίζει τον αριθμό των συστάδων στις οποίες επιθυμούμε να ομαδοποιηθούν τα δεδομένα μας. Όταν η τιμή είναι -1, τότε ο αλγόριθμος καθορίζει τον αριθμό των συστάδων με την διαδικασία Cross Validation.

- **numFolds:** Με την παράμετρο αυτή καθορίζουμε τον αριθμό των σημείων που θα χρησιμοποιηθούν κατά την διαδικασία Cross Validation για καλύτερη εύρεση συστάδων (clusters). Η αρχική τιμή είναι 10, εκτός το σύνολο των δεδομένων που θα χρησιμοποιήσουμε είναι μικρότερος του 10 οπότε στην περίπτωση αυτή η τιμή της παραμέτρου ορίζεται ίση με το σύνολο των εγγραφών που περιέχονται στο αρχείο των δεδομένων.

- **maxIterations:** Η παράμετρος αυτή καθορίζει τον μέγιστο αριθμό των επαναλήψεων που μπορεί να εκτελέσει ο αλγόριθμος κατά την ομαδοποίηση.

- **numExecutionSlots:** Η τιμή της παραμέτρου αυτής καθορίζει τον αριθμό των επεξεργαστών που θα χρησιμοποιήσει το WEKA για την εκτέλεση του αλγορίθμου. Η τιμή αυτή εξαρτάται από τα χαρακτηριστικά του εκάστοτε υπολογιστή στον οποίο εκτελείται η διαδικασία συσταδοποίησης. Συνήθως η τιμή της παραμέτρου ορίζεται ίση με τον αριθμό των διαθέσιμων πυρήνων (cpu/cores)..

- **displayModelInOldFormat:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, τα αποτελέσματα του αλγορίθμου εξάγονται σε παλιά μορφή (βλέπε εικόνα3). Η μορφή αυτή είναι καταλληλότερη όταν δημιουργούνται πολλές ομάδες. Επιλέγοντας την τιμή β)Ψευδής λαμβάνουμε τα αποτελέσματα σε νέα μορφή (βλέπε εικόνα4) που είναι προτιμότερη όταν θέλουμε να ομαδοποιήσουμε αρχείο με πολλά χαρακτηριστικά και ο αριθμός των συστάδων που θα δημιουργηθεί είναι μικρός.



Attribute	Cluster 0 (0.36)	1 (0.13)
age		
mean	43.6697	42.9635
std. dev.	11.8291	12.0568

(α)

```
Cluster: 0 Prior probability: 0.3611

Attribute: age
Normal Distribution. Mean = 43.6697 StdDev = 11.8291


Cluster: 1 Prior probability: 0.1344

Attribute: age
Normal Distribution. Mean = 42.9635 StdDev = 12.0568
```

(β)

Εικόνα 3.13: Απεικόνιση αποτελεσμάτων αλγορίθμου. (α) Παλιά Μορφή Εξόδου Αποτελεσμάτων, (β) Νέα Μορφή Εξόδου Αποτελεσμάτων

- **maximumNumberOfClusters:** Η παράμετρος αυτή καθορίζει τον μέγιστο αριθμό συστάδων (clusters) που μπορεί να δημιουργηθεί κατά την διαδικασία Cross Validation ώστε να λάβουμε καλύτερο αποτέλεσμα κατά την ομαδοποίηση του αλγορίθμου.

Τα αποτελέσματα εξόδου του αλγορίθμου EM όπως παρουσιάζονται στο WEKA.

Τα αποτελέσματα εξόδου του αλγορίθμου EM στο σύστημα WEKA είναι τα εξής:

- Ο αριθμός των συστάδων που δημιουργήθηκαν.
- Ο αριθμός των επαναλήψεων που πραγματοποίησε ο αλγόριθμος.
- Η μέση τιμή και η τυπική απόκλιση του κάθε χαρακτηριστικού για κάθε συστάδα.
- Ο χρόνος ολοκλήρωσης της συσταδοποίησης.

- Το σύνολο και το ποσοστό των εγγραφών για κάθε συστάδα.
- Η λογαριθμική πιθανοφάνεια (log likelihood) που προκύπτει από την συσταδοποίηση.

3.1.6 Αλγόριθμος FartherFirst

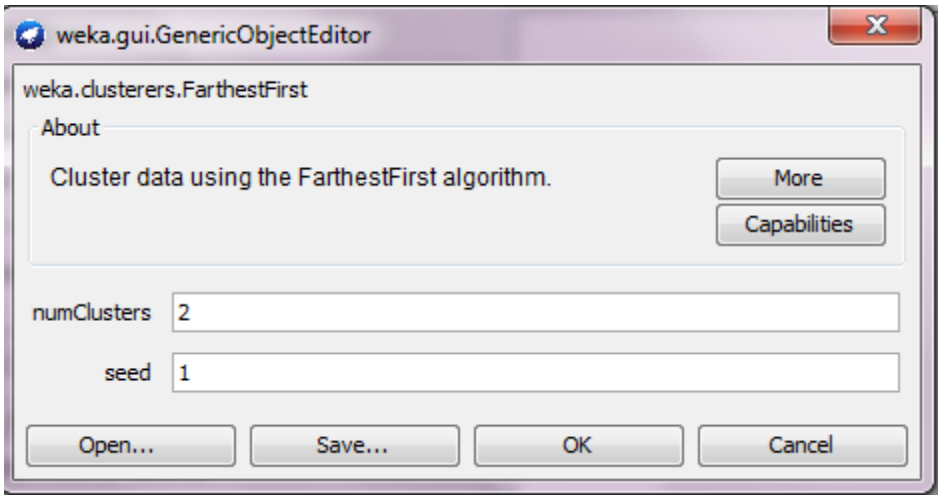
Ο αλγόριθμος Farthest First [34] αποτελεί μια παραλλαγή του αλγορίθμου K-Mean η οποία βελτιώνει τον χρόνο που απαιτείται για την εύρεση των K συστάδων μέσα στο σύνολο των δεδομένων μας.

Σε αντίθεση με τον αλγόριθμο K-Means, ο οποίος επαναπροσδιορίζει τη θέση του κέντρου της κάθε κλάσης στο σημείο το οποίο ελαχιστοποιεί το άθροισμα των αποστάσεων μεταξύ αυτού και των σημείων που ανήκουν στην κλάση, ο αλγόριθμος Farther First επανατοποθετεί το κάθε κέντρο στο σημείο της ομάδας το οποίο απέχει τη μεγαλύτερη απόσταση από το προηγούμενο κέντρο.

Με τη χρήση αυτής της μεθόδου έχει αποδειχθεί (με ευριστικό τρόπο) ότι επιταχύνεται σημαντικά ο χρόνος σύγκλισης που απαιτείται για την εύρεση των κέντρων των K ομάδων. Αυτή η επιτάχυνση μπορεί να αιτιολογηθεί από το γεγονός ότι ο αλγόριθμος αυτός δεν απαιτεί τον υπολογισμό της θέσης του κέντρου, μιας και κάθε φορά χρησιμοποιείται ως κέντρο της ομάδας, ένα από τα σημεία του συνόλου δεδομένων μας.

Παράμετροι εισόδου του αλγορίθμου FarthestFirst στο WEKA

Ο αλγόριθμος FarthestFirst έχει πολυπλοκότητα O(nk), όπου n = αριθμός σημείων, K = αριθμός συστάδων.



Εικόνα 3.14: Οι παράμετροι εισόδου του αλγορίθμου FarthestFirst στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο FarthestFirst στο WEKA περιγράφονται παρακάτω.

- **seed:** Η παράμετρος αυτή καθορίζει τον αριθμό των αρχικών κέντρων που θα χρησιμοποιηθούν κατά την εκτέλεση του αλγορίθμου.
- **numClusters:** Η παράμετρος αυτή καθορίζει τον αριθμό των ομάδων που θα δημιουργήσει ο αλγόριθμος.

Τα αποτελέσματα εξόδου του αλγορίθμου FarthestFirst όπως παρουσιάζονται στο WEKA.

Τα αποτελέσματα εξόδου του αλγορίθμου FarthestFirst είναι τα εξής:

- Οι τιμές των χαρακτηριστικών των κέντρων κάθε συστάδας.
- Το σύνολο και το ποσοστό των εγγραφών κάθε συστάδας.

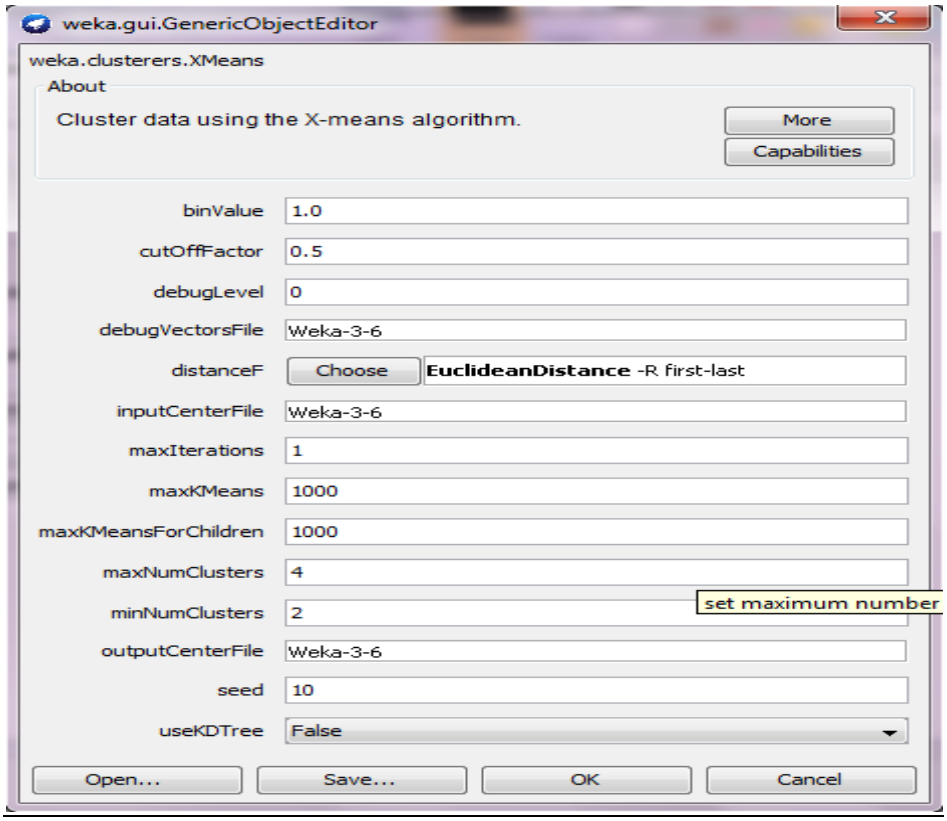
3.1.7 Αλγόριθμος X-Means

Ο διαμεριστικός αλγόριθμος X-Means [35] είναι μια παραλλαγή του αλγορίθμου SimpleKMeans η οποία προσπαθεί να επιλύσει το βασικότερο μειονέκτημα του. Όπως αναφέραμε παραπάνω ο αλγόριθμος SimpleKMeans απαιτεί σαν είσοδο τον αριθμό των συστάδων που σχηματίζονται από τα δεδομένα, γεγονός το οποίο περιορίζει τις εφαρμογές του μιας και στις περισσότερες περιπτώσεις ο αριθμός αυτός δεν είναι γνωστός.

Ο αλγόριθμος X-Means αντιμετωπίζει τα προβλήματα αυτά με τη χρήση κάποιων στατιστικών κριτηρίων (BIC – Bayesian Information Criterion, AIC – Akaike Information Criterion) τα οποία μας βοηθούν, χρησιμοποιώντας τα δεδομένα μας, να καθορίσουμε με γρήγορο και αποδοτικό τρόπο, τον βέλτιστο αριθμό των κλάσεων που μπορούν να δημιουργηθούν από αυτά.

Παράμετροι εισόδου του αλγορίθμου X-Means στο WEKA

Ο αλγόριθμος X-Means έχει πολυπλοκότητα O(n), όπου n = αριθμός σημείων.



Εικόνα 3.15: Οι παράμετροι εισόδου του αλγορίθμου X-Means στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο X-Means στο WEKA περιγράφονται παρακάτω.

- **binValue:** Η παράμετρος αυτή ορίζει την τιμή που αντιπροσωπεύει τα νέα χαρακτηριστικά.
- **cutOffFactor:** Με την τιμή της παραμέτρου αυτής ορίζεται το κατώφλι που θα χρησιμοποιηθεί από τον αλγόριθμο κατά την εκτέλεσή του.
- **debugVectorsFile:** Η παράμετρος αυτή λαμβάνει ως είσοδο το αρχείο το οποίο χρησιμοποιείται για αποσφαλμάτωση.
- **distanceF**: Στην παράμετρο αυτή επιλέγεται η συνάρτηση απόστασης που θα χρησιμοποιήσει ο αλγόριθμος.
- **inputCenterFile:** Η παράμετρος αυτή λαμβάνει ως είσοδο το αρχείο από το οποίο θα διαβάσει την λίστα με τα κέντρα.

- **maxIterations:** Με την τιμή της παραμέτρου ορίζεται ο μέγιστος αριθμός επαναλήψεων που θα εκτελέσει ο αλγόριθμος.
- **maxKMeans**: Με την τιμή της παραμέτρου ορίζεται ο μέγιστος αριθμός επαναλήψεων που θα εκτελέσει ο αλγόριθμος KMeans.
- **maxKMeansForChildren:** Με την τιμή της παραμέτρου ορίζεται ο μέγιστος αριθμός επαναλήψεων που θα εκτελεστούν για την εύρεση των αρχικών κέντρων.
- **maxNumClusters:** Η παράμετρος αυτή λαμβάνει ως είσοδο τον μέγιστο αριθμό των συστάδων που θα δημιουργηθούν.
- **minNumClusters:** Η παράμετρος αυτή λαμβάνει ως είσοδο τον ελάχιστο αριθμό των συστάδων που θα δημιουργηθούν.
- **outputCenterFile**: Η παράμετρος αυτή λαμβάνει ως είσοδο το αρχείο στο οποίο θα γραφτούν τα κέντρα.
- **seed:** Η τιμή της παραμέτρου αυτής καθορίζει τον αριθμό των αρχικών κέντρων. Η επιλογή της τιμής είναι τυχαία.
- **useKDTree:** Η παράμετρος αυτή λαμβάνει δύο διακριτές τιμές α) Αληθής (True) και β) Ψευδής (False). Με την επιλογή της τιμής α) Αληθής, γίνεται η χρήση KDTree ενώ με την β) Ψευδής ο αλγόριθμος δεν χρησιμοποιεί KDTree.

Τα αποτελέσματα εξόδου του αλγορίθμου X-Means όπως παρουσιάζονται στο WEKA.

Τα αποτελέσματα εξόδου του αλγορίθμου X-Means είναι τα εξής:

- Οι τιμές των χαρακτηριστικών των κέντρων κάθε ομάδας.
- Το σύνολο και το ποσοστό των εγγραφών κάθε ομάδας.
- Η τιμή του BIC κριτηρίου με το οποίο επιλέχθηκε ο αριθμός των συστάδων.

3.1.8 Αλγόριθμος OPTICS (Ordering Points To Identify the Clustering Structure)

Ο αλγόριθμος OPTICS [36] όπως ο DBSCAN είναι ένας αλγόριθμος ο οποίος βασίζεται στην πυκνότητα και σκοπός του είναι η ανακάλυψη συστάδων σε χωρικά δεδομένα.

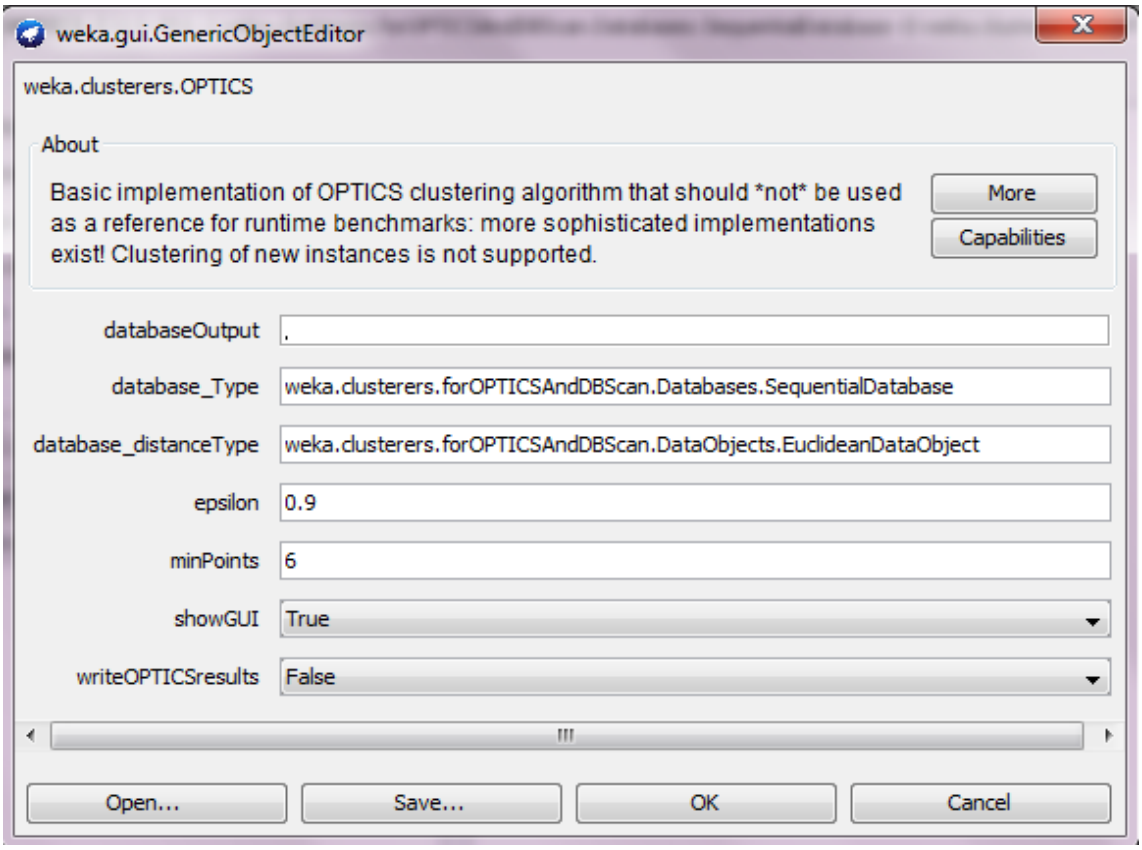
Η ιδέα του OPTICS είναι παρόμοια με του DBSCAN αλλά αντιμετωπίζει ένα σοβαρό μειονέκτημα του DBSCAN, που έγκειται στο πρόβλημα της ανίχνευσης συστάδων σε δεδομένα μεταβλητής πυκνότητας. Για να το επιτύχει αυτό χρησιμοποιεί τεχνικές συσταδοποίησης των δεδομένων έτσι ώστε αυτά που είναι χωρικά πιο κοντά, να γίνονται γείτονες. Επιπλέον μια ειδική απόσταση αποθηκεύεται, για κάθε δεδομένο που αναπαριστά την πυκνότητα που χρειάζεται, ώστε να αποτελέσει μέρος μιας ομάδας.

Ο OPTICS όπως και ο DBSCAN απαιτεί την εισαγωγή δύο παραμέτρων, της ελάχιστης ακτίνας (*Eps*) και του ελάχιστου αριθμού σημείων (*MinPts*) και παράγει ένα ειδικού τύπου δενδροδιάγραμμα από το οποίο εξάγεται η ιεραρχική δομή των κλάσεων. Η πολυπλοκότητα του αλγορίθμου είναι O (*n* log *n*).

Επίσης ο OPTICS σε αντίθεση με τον DBSCAN θεωρεί επίσης σημεία που είναι μέλη μιας πιο πυκνής κλάσης, έτσι ώστε σε κάθε σημείο να προσδίδεται μια απόσταση πυρήνα όπου περιγράφει την απόσταση του πλησιέστερου από τα MinPts.

Παράμετροι εισόδου του αλγορίθμου OPTICS στο WEKA

Ο αλγόριθμος OPTICS έχει πολυπλοκότητα O(n log n), όπου n = αριθμός σημείων.



Εικόνα 3.16: Οι παράμετροι εισόδου του αλγορίθμου OPTICS στο σύστημα WEKA

Οι παράμετροι εισόδου που δέχεται ο OPTICS στο WEKA περιγράφονται παρακάτω.

- **databaseOutput:** Στην παράμετρο αυτή ορίζεται το αρχείο για την αποθήκευση του παραγόμενου αρχείου. Η τιμή αυτή είναι προαιρετική και χρησιμεύει στην οπτικοποίηση των δεδομένων.
- **database_Type:** Η παράμετρος αυτή καθορίζει τον τύπο της βάσης που θα χρησιμοποιηθεί κατά την εκτέλεση του αλγορίθμου για την αποθήκευση του αρχικού αρχείου δεδομένων.
- **database_distanceType:** Η τιμή αυτής της παραμέτρου καθορίζει την μετρική αποστάσεων που θα χρησιμοποιηθεί από τον αλγόριθμο όπως για παράδειγμα η Ευκλείδεια.
- **epsilon:** Με την παράμετρο αυτή ορίζουμε την ελάχιστη ακτίνα που καθορίσει την περιοχή των σημείων τα οποία θα ελεγχθούν κατά την εκκίνηση της διαδικασίας συσταδοποίησης.
- **minPoints:** Η τιμή αυτής της παραμέτρου ορίζει τον ελάχιστο αριθμό σημείων που απαιτείται να περιέχονται μέσα την περιοχή ενός σημείου (πυρήνα).
- **showGUI:** Η τιμή της παραμέτρου αυτής καθορίζει το αν τα αποτελέσματα της συσταδοποίησης του αλγορίθμου θα οπτικοποιηθούν μετά την εκτέλεση του ή όχι.
- **writeOPTICSresults:** Αν η τιμή της παραμέτρου είναι Ψευδής τότε τα αποτελέσματα αποθηκεύονται στο αρχείο με το όνομα OPTICS_#TimeStamp#.TXT

Τα αποτελέσματα εξόδου του αλγορίθμου OPTICS όπως παρουσιάζονται στο WEKA.

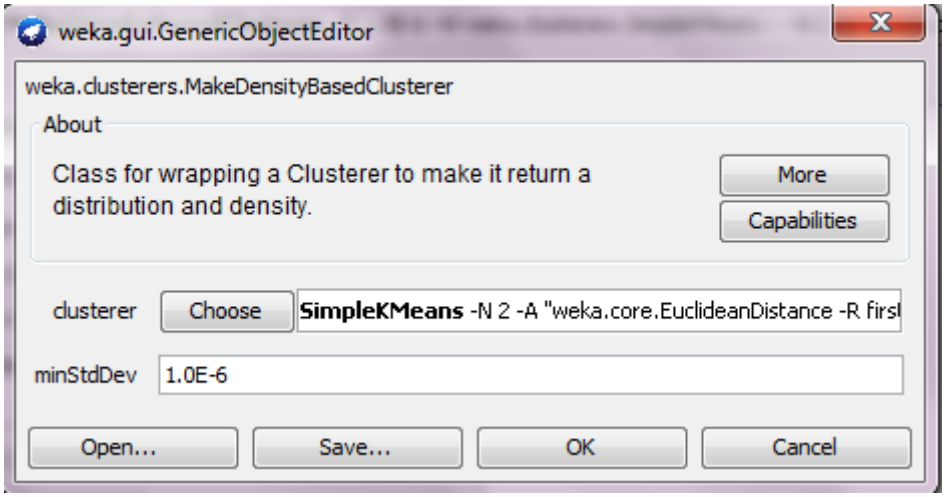
Τα αποτελέσματα εξόδου του αλγορίθμου OPTICS είναι τα εξής:

- Το σύνολο των εγγραφών που δεν εντάχθηκαν σε καμία συστάδα.
- Οι τιμές των χαρακτηριστικών κάθε εγγραφής που δεν ανήκει σε καμία συστάδα.

3.1.9 Αλγόριθμος MakeDensityBasedClusterer

Με την επιλογή του MakeDensityBasedClusterer, μπορούμε να χρησιμοποιήσουμε οποιονδήποτε από τους προαναφερθέντες αλγορίθμους οι οποίοι στην περίπτωση αυτή, λειτουργούν με την εξής λογική. Συμπληρώνοντας την επιθυμητή τιμή στην παράμετρο της ελάχιστη επιτρεπτή τυπικής απόκλισης (minStdDev), ο αλγόριθμος μας επιστρέφει την πιθανοφάνεια των παραμέτρων ενός μοντέλου κανονικής κατανομής που ταιριάζουν βέλτιστα στα δεδομένα.

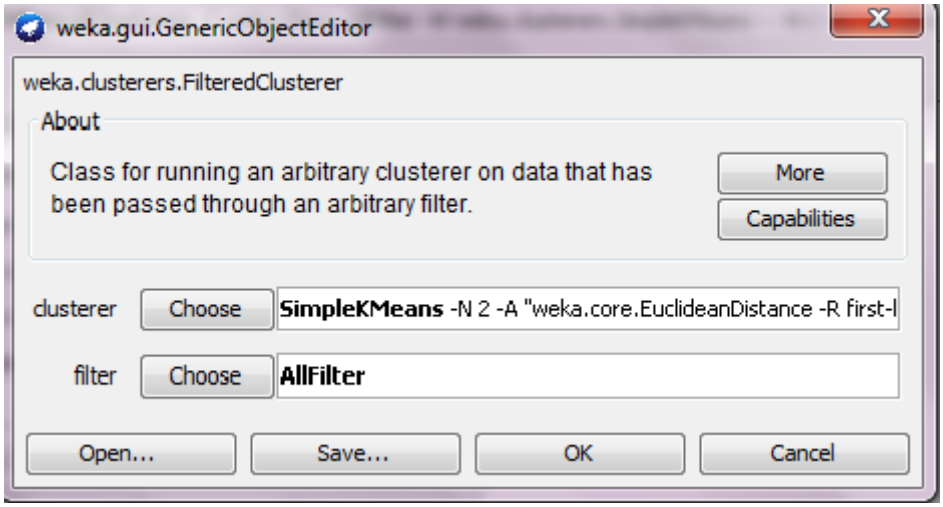
Οι παράμετροι εισόδου φαίνονται στην εικόνα 3.17. Τα αποτελέσματα εξόδου παράγονται ανάλογα με την επιλογή του αλγορίθμου.



Εικόνα 3.17: Οι παράμετροι εισόδου του αλγορίθμου MakeDensityBasedClusterer στο σύστημα WEKA

3.1.10 Αλγόριθμος FilteredClusterer

Με την επιλογή του FilteredClusterer, μπορούμε να χρησιμοποιήσουμε οποιονδήποτε από τους προαναφερθέντες αλγορίθμους οι οποίοι πλέον εκτελούνται εφόσον έχει προηγηθεί φιλτράρισμα των δεδομένων στα οποία θα εκτελεστεί ο αλγόριθμος, με βάση το εκάστοτε επιλεγμένο φίλτρο. Οι παράμετροι εισόδου φαίνονται στην εικόνα 3.18. Τα αποτελέσματα εξόδου παράγονται ανάλογα με την επιλογή του αλγορίθμου.



Εικόνα 3.18: Οι παράμετροι εισόδου του αλγορίθμου FilteredClusterer στο σύστημα WEKA

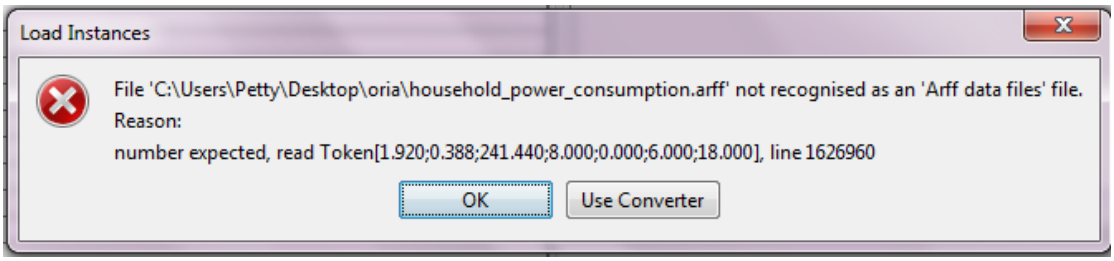
Κεφάλαιο 4 - Σύγκριση των Αλγορίθμων του WEKA

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τα αποτελέσματα των αλγορίθμων του WEKA τα οποία έχουν ως σκοπό να δείξουν τα όρια του κάθε αλγορίθμου, όταν χρησιμοποιούνται αρχεία δεδομένων με μεγάλο αριθμό εγγραφών (εκατοντάδες χιλιάδες) και χαρακτηριστικών.

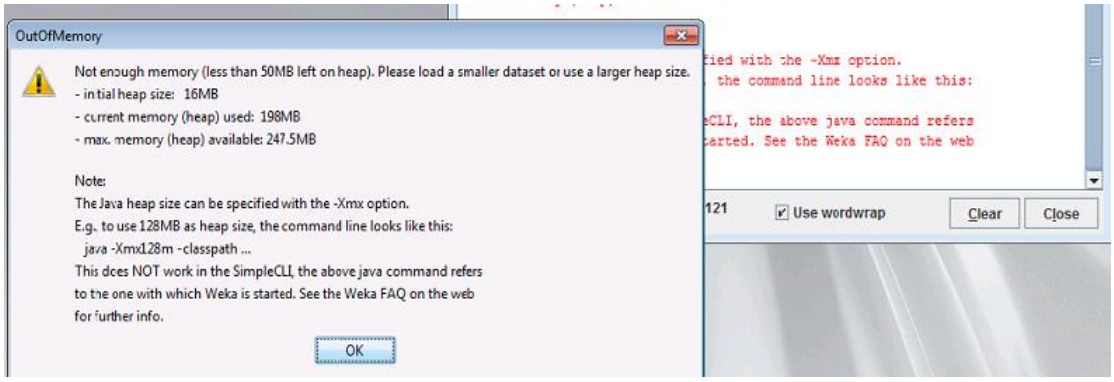
4.1 Προετοιμασία του WEKA για διαχείριση μεγάλων αρχείων

Το σύστημα WEKA, κατόπιν πειραμάτων που πραγματοποιήθηκαν, παρατηρήσαμε ότι δεν μπορεί να διαχειριστεί μεγάλα αρχεία δεδομένων λόγω σφαλμάτων διαχείρισης μνήμης που παρουσίασε. Τα σφάλματα που παρουσιάζονται είναι τα εξής:

- κατά την φόρτωση του αρχείου στο WEKA (βλέπε εικόνα 4.1)
- κατά την εκτέλεση των αλγορίθμων (βλέπε εικόνα 4.2).



Εικόνα 4.1: Σφάλμα κατά την φόρτωση αρχείου δεδομένων



Εικόνα 4.2: Σφάλμα κατά την διαδικασία συσταδοποίησης

Μετά από έρευνα που πραγματοποιήσαμε, διαπιστώθηκε ότι αυτό το πρόβλημα μπορούσε να βελτιωθεί αλλάζοντας τις αρχικές ρυθμίσεις του WEKA. Παρακάτω θα παρουσιάσουμε τα βήματα με τα οποία μπορούν να αλλαχθούν οι αρχικές ρυθμίσεις ώστε να ξεπεραστεί το πρόβλημα ανεπάρκειας της μνήμης:

Στο μονοπάτι (path) εγκατάστασης του WEKA, υπάρχουν τα αρχεία RunWEKA.ini και RunWEKA.bat. Στο πρώτο αρχείο βρίσκουμε την παράμετρο maxheap και αυξάνουμε την τιμή της. Για τις ανάγκες της παρούσας διπλωματικής και βάσει των τεχνικών χαρακτηριστικών του υπολογιστή στον οποίο εκτελέστηκαν τα πειράματα, αυξήσαμε την τιμή της παραμέτρου maxheap σε 1024M (όπου ήταν και η μέγιστη δυνατή). Στο δεύτερο αρχείο βρίσκουμε την εντολή %_java% -Xmx1024m -classpath . RunWEKA -i .\RunWEKA.ini -w .\WEKA.jar -c %_cmd% "%2" και αλλάζουμε την τιμή της παραμέτρου -Xmx ίση με την τιμή της παραμέτρου maxheap.

4.2 Παρουσίαση αποτελέσματα των αλγορίθμων του WEKA

Προτού προχωρήσουμε στην παρουσίαση των αποτελεσμάτων των αλγορίθμων του WEKA, θα αναφερθούμε στα τεχνικά χαρακτηριστικά του συστήματος (H/Y, έκδοση λογισμικών) το οποίο χρησιμοποιήθηκε για την εκτέλεση των πειραμάτων. Τα τεχνικά χαρακτηριστικά φαίνονται στον παρακάτω πίνακα:

Τεχνικά χαρακτηριστικά H/Y - WEKA	
Memory(RAM)	6,00GB DDR3 1066MHz
Processor	Intel Core i5-2450M CPU 2,5GHz
System Type	64-bit Operating System
Windows edition	Windows 7 Home Premium
WEKA Version	3.6.12

Πίνακας 4.1: Τεχνικά χαρακτηριστικά συστήματος εκτέλεσης αλγορίθμων WEKA

Για την αξιολόγηση και σύγκριση των αλγορίθμων, χρησιμοποιήθηκε ως βασικό σύνολο δεδομένων το αρχείο "USCensus1990" το οποίο προήλθε από τη βάση δεδομένων του Πανεπιστημίου Irvine, California (UCI Machine Learning Repository). Επιλέξαμε το συγκεκριμένο αρχείο διότι περιλαμβάνει μεγάλο αριθμό εγγραφών και μεγάλο αριθμό χαρακτηριστικών, κάτι το οποίο επηρεάζει σημαντικά την απόδοση των αλγορίθμων του WEKA. Πιο συγκεκριμένα το αρχείο περιλαμβάνει 2.458.285 εγγραφές με 68 χαρακτηριστικά (βλέπε πίνακα 4.2). Αξίζει να σημειωθεί ότι αυτό το αρχείο ήταν από τα λίγα διαθέσιμα, τα οποία συνδύαζαν και μεγάλο αριθμό εγγραφών αλλά και χαρακτηριστικών. Παρόλα αυτά το μέγεθος του συγκεκριμένου αρχείου δεν μπορούσε να φορτωθεί στο WEKA, και επομένως αναγκαστήκαμε να καταφύγουμε σε μέθοδο τεμαχισμού αυτού του αρχείου σε μικρότερα.

Αρχείο δεδομένων USCensus1990	
Records	2.458.285
Attributes	68

Πίνακας 4.2: Περιέχει τον σύνολο των εγγραφών και το σύνολο του αριθμό των χαρακτηριστικών του αρχείου USCensus1990

Κατόπιν πειραμάτων, το μεγαλύτερο αρχείο που μπορέσαμε να φορτώσουμε στο WEKA το οποίο συνδυάζει και μεγάλο αριθμό εγγραφών αλλά και χαρακτηριστικών ήταν 400.000 εγγραφές με 50 χαρακτηριστικά. Αξίζει να σημειωθεί ότι τα όρια σε σχέση με τον αριθμό των εγγραφών που μπορούν να φορτωθούν στο WEKA, αλλάζουν σε συνάρτηση με τον αριθμό χαρακτηριστικών. Για παράδειγμα σε περίπτωση που χρησιμοποιήσουμε μόνο 2 χαρακτηριστικά το αρχείο μπορεί να φορτωθεί ολόκληρο.

Προκειμένου να παρουσιάσουμε τα όρια εκτέλεσης κάθε αλγορίθμου, δημιουργήσαμε μια βιβλιοθήκη αρχείων χρησιμοποιώντας το αρχείο "USCensus1990". Δημιουργήθηκαν αρχεία με αριθμό εγγραφών {100.000, 200.000, 300.000, 400.000} και για κάθε αριθμό εγγραφών, διαφορετικός αριθμός χαρακτηριστικών {10, 20, 30, 40, 60} για τους τρεις πρώτους, και {10, 20, 30, 40, 50} για αριθμό εγγραφών 400.000. Η επιλογή των εγγραφών αλλά και των χαρακτηριστικών έγινε με τυχαίο τρόπο.

Για την εκτέλεση των πειραμάτων στο WEKA χρησιμοποιήσαμε την λειτουργία συσταδοποίησης (cluster mode) "Use Training set". Παρακάτω παρουσιάζουμε τα αποτελέσματα και τα όρια του κάθε αλγορίθμου:

4.2.1 Αποτελέσματα Αλγορίθμου SimpleKMeans

Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

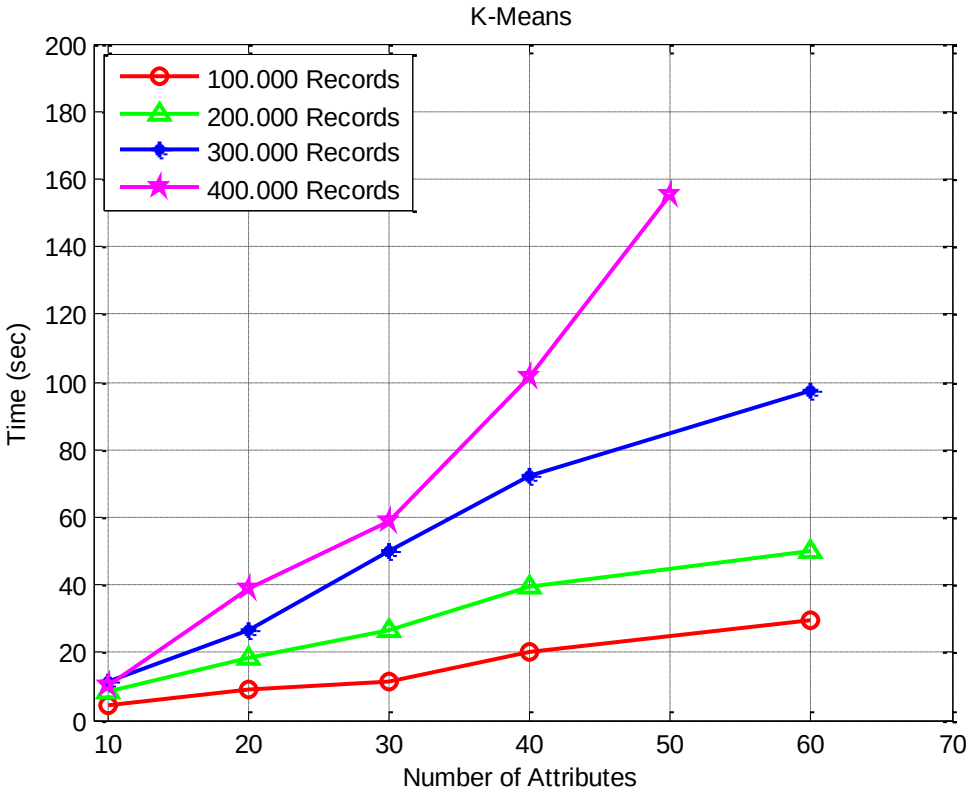
- displayStdDevs: False
- distanceFunction: EuclideanDistance (Default)
- dontReplaceMissingValues: False
- maxIterations: 500

- numClusters: 4
- preserveInstancesOrder: False
- seed: 10

Στον πίνακα 4.3 φαίνονται για κάθε αριθμό εγγραφών, οι χρόνοι εκτέλεσης του αλγορίθμου SimpleKMeans συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.3.

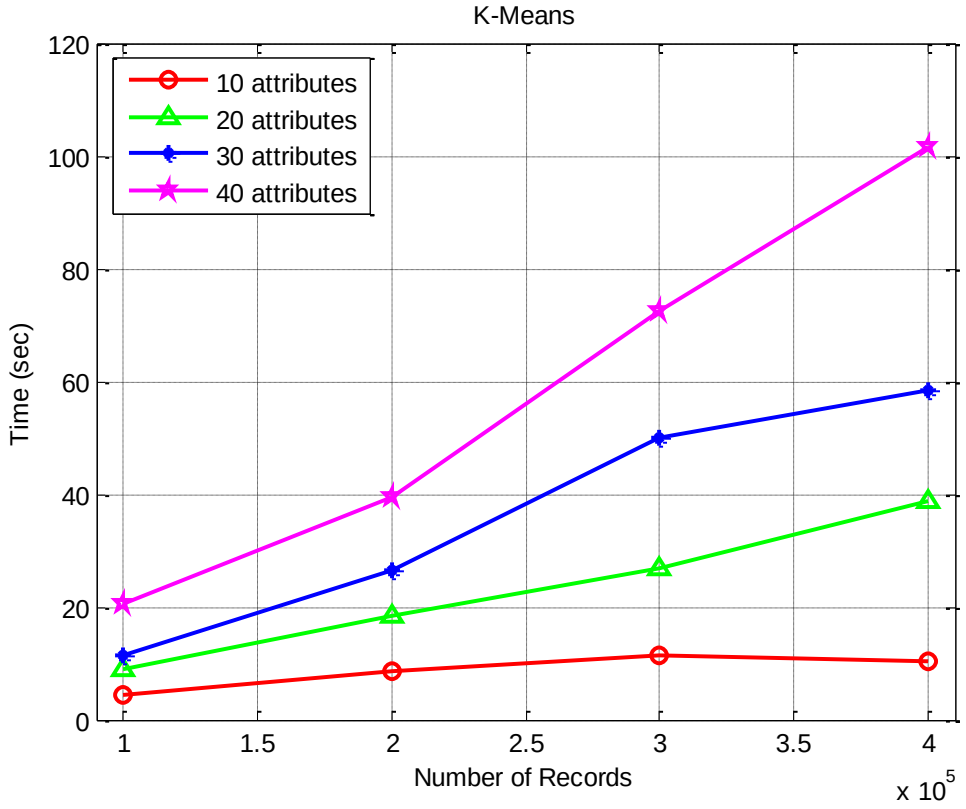
Πίνακας Αποτελεσμάτων αλγορίθμου SimpleKMeans - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	4.46	9.06	11.44	20.37	29.45
200.000	8.52	18.49	26.62	39.33	50.17
300.000	11.5	26.88	50.09	72.32	97.36
400.000	10.41	38.86	58.57	101.66	155.47

Πίνακας 4.3: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών



Εικόνα 4.3: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του SimpleKMeans συναρτήσει του αριθμού των χαρακτηριστικών

Όπως παρατηρούμε από το διάγραμμα της εικόνας 4.3, ο SimpleKMeans πετυχαίνει πολύ μικρούς χρόνους εκτέλεσης λόγω της χαμηλής πολυπλοκότητάς του (βλέπε κεφάλαιο 3) . Επίσης παρατηρείται ότι για κάθε τιμή εγγραφών, ο χρόνος εκτέλεσης του αλγορίθμου (εκτός της περίπτωσης των 400.000 εγγραφών) αυξάνει σχεδόν γραμμικά με την αύξηση του αριθμού των χαρακτηριστικών.



Εικόνα 4.4: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του SimpleKMeans συναρτήσει του αριθμού των εγγραφών.

Στο διάγραμμα της εικόνας 4.4 παρατηρούμε ότι για κάθε τιμή χαρακτηριστικών, ο χρόνος εκτέλεσης του αλγορίθμου αυξάνει σχεδόν γραμμικά με την αύξηση του αριθμού των εγγραφών. Μια ενδιαφέρουσα παρατήρηση είναι ότι καθώς αυξάνεται ο αριθμός των εγγραφών, η διαφορά των χρόνων εκτέλεσης για διαφορετικές τιμές του αριθμού χαρακτηριστικών αυξάνεται σημαντικά.

4.2.2 Αποτελέσματα Αλγορίθμου XMeans

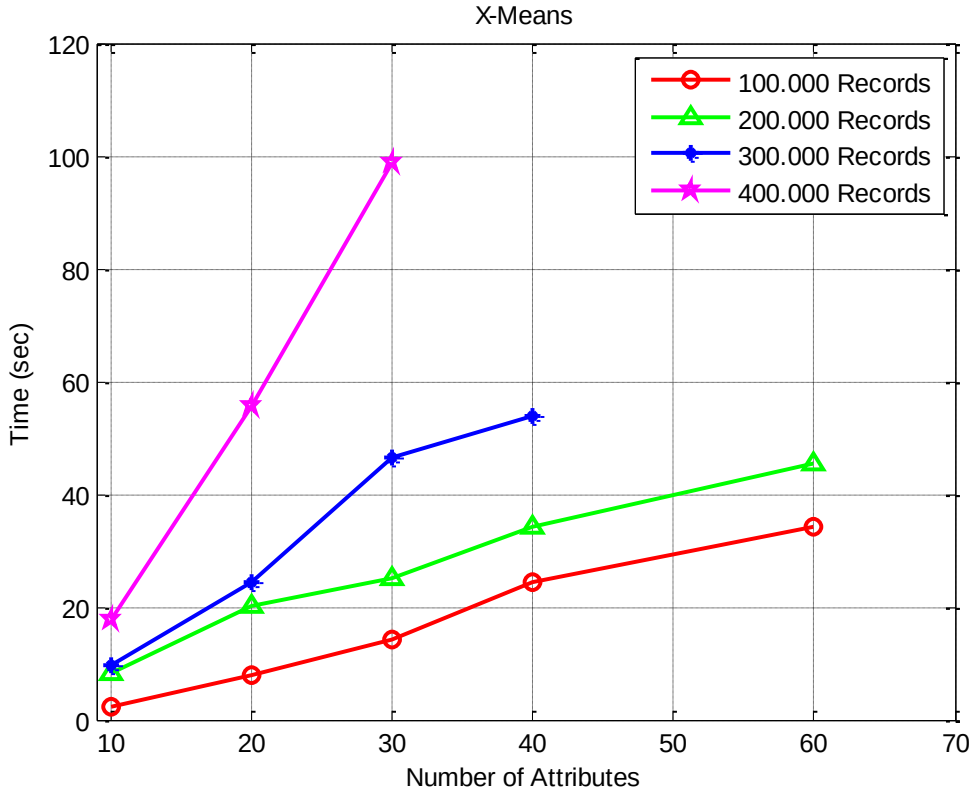
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- binValue: 1.0
- cutOffFactor: 0.5
- debugLevel: 0
- debugVectorsFile: WEKA-3-6
- distanceF: EuclideianDistance (Default)
- inputCenterFile: WEKA-3-6
- maxIterations: 1
- maxKMeans: 1000
- maxKMeansForChildren: 1000
- maxNumClusters: 4
- minNumClusters: 4
- outputCenterFile: WEKA-3-6
- seed: 10
- useKDTree: False

Στον πίνακα 4.4 φαίνονται για κάθε αριθμό εγγραφών, οι χρόνοι εκτέλεσης του αλγορίθμου Xmeans συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.5.

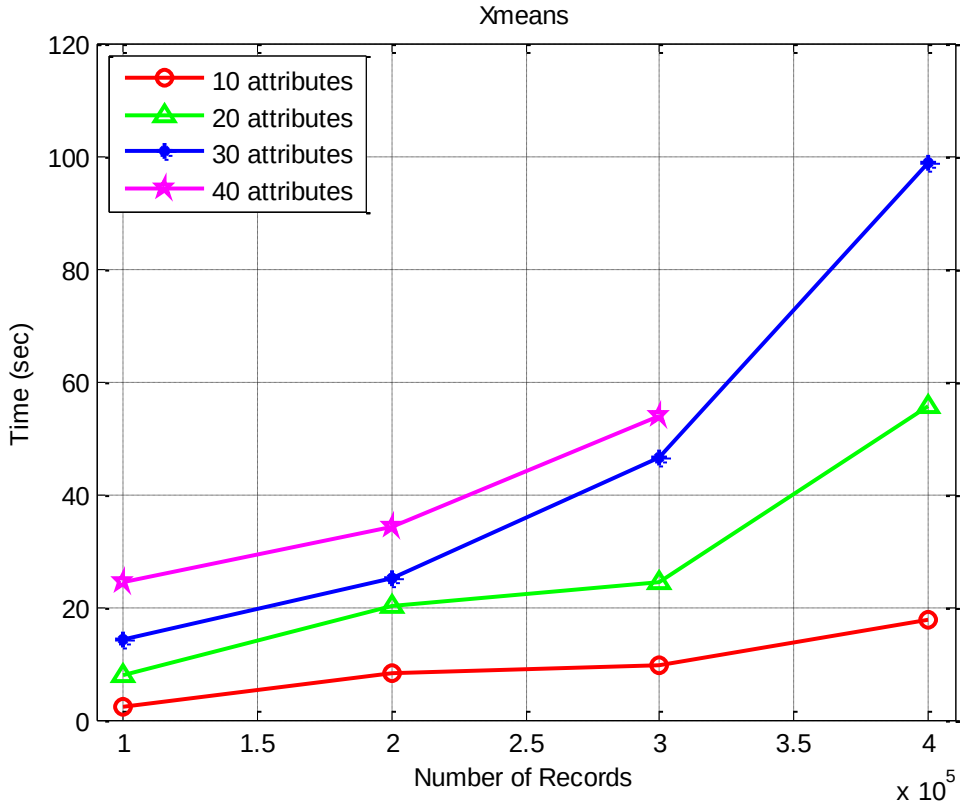
Πίνακας Αποτελεσμάτων αλγορίθμου Xmeans - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	2.45	7.74	14.31	24.48	34.07
200.000	8.25	20.09	25.12	34.14	45.55
300.000	9.71	24.42	46.64	53.93	Nan
400.000	17.67	55.71	98.92	Nan	Nan

Πίνακας 4.4: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.5: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του XMeans συναρτήσει του αριθμού των χαρακτηριστικών.

Όπως παρατηρούμε από το διάγραμμα της εικόνας 4.5, ο Xmeans πετυχαίνει πολύ μικρούς χρόνους εκτέλεσης λόγω της χαμηλής πολυπλοκότητάς του (βλέπε κεφάλαιο 3) . Για τους αριθμούς εγγραφών 100.000 και 200.000, οι χρόνοι εκτέλεσης των αλγορίθμων αυξάνονται γραμμικά και με μικρό ρυθμό αύξησης συναρτήσει του αριθμού των χαρακτηριστικών. Όμως για τους αριθμούς εγγραφών 300.000 και 400.000, ο ρυθμός αύξησης μεγαλώνει (μεγαλύτερη κλίση των ευθειών) και παρατηρείται αδυναμία ολοκλήρωσης εκτέλεσης του αλγορίθμου, όταν ο αριθμός των χαρακτηριστικών ξεπεράσει τα 40 και 30 αντίστοιχα. Η αδυναμία εκτέλεσης του αλγορίθμου οφείλεται στην έλλειψη διαθέσιμης μνήμης του συστήματος.



Εικόνα 4.6: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του Xmeans συναρτήσει του αριθμού των εγγραφών.

Στο διάγραμμα της εικόνας 4.6, παρατηρούμε για κάθε τιμή χαρακτηριστικών την μεταβολή των χρόνων εκτέλεσης του αλγορίθμου συναρτήσει του αριθμού των εγγραφών. Όπως φαίνεται, όσο αυξάνεται ο αριθμός των χαρακτηριστικών και των εγγραφών, ο χρόνος εκτέλεσης του αλγορίθμου αυξάνει σημαντικά και ιδιαίτερα όταν έχουμε μεγάλες τιμές αριθμού εγγραφών.

4.2.3 Αποτελέσματα Αλγορίθμου MakeDensity

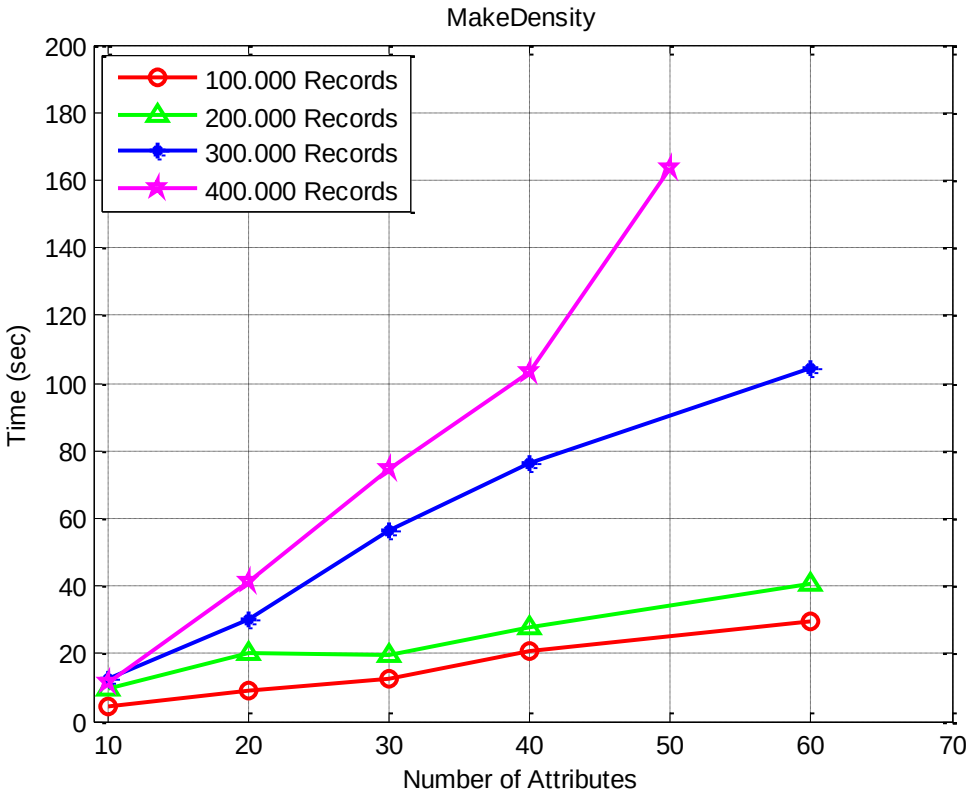
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- clusterer: SimpleKMeans (βλέπε τις παραμέτρους του SimpleKMeans)
- minStdDev: 1.0E-6

Στον πίνακα 4.5 φαίνονται για κάθε αριθμό εγγραφών, οι χρόνοι εκτέλεσης του αλγορίθμου MakeDensity συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.7.

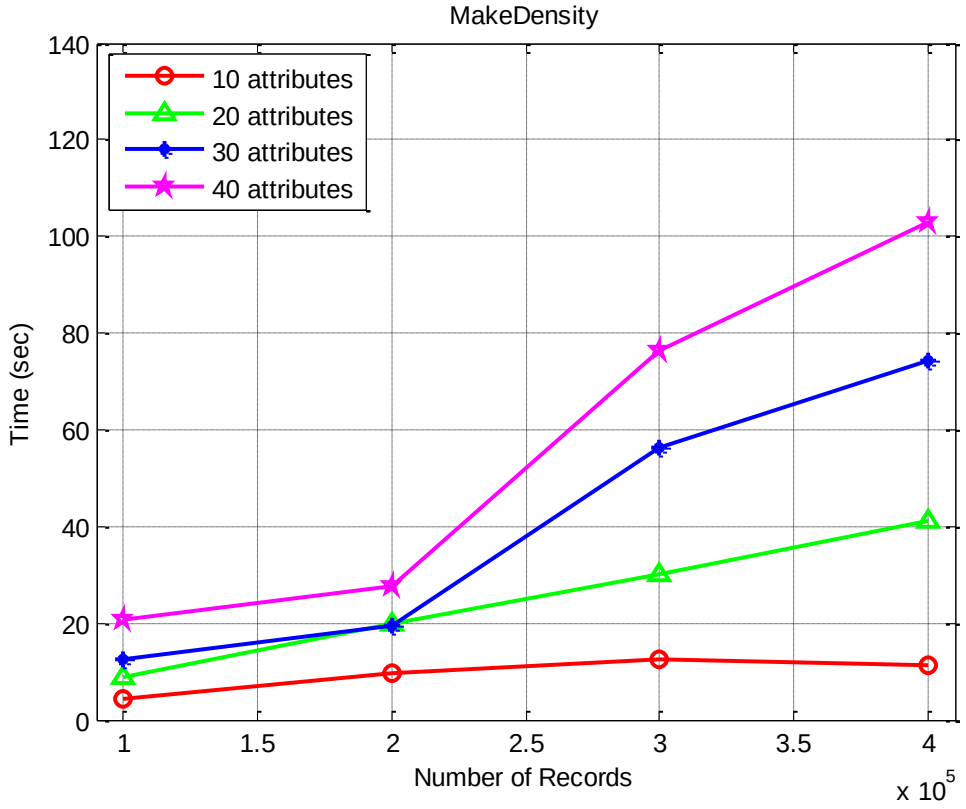
Πίνακας Αποτελεσμάτων αλγορίθμου MakeDensity - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	4.4	8.85	12.29	20.62	29.36
200.000	9.66	20.33	19.32	27.59	40.77
300.000	12.57	30.22	56.35	76.22	104.66
400.000	11.4	41.11	74.36	103.01	163.16

Πίνακας 4.5: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.7: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του MakeDensity συναρτήσει του αριθμού των χαρακτηριστικών.

Όπως παρατηρούμε από το διάγραμμα της εικόνας 4.7, ο MakeDensity πετυχαίνει πολύ μικρούς χρόνους εκτέλεσης λόγω της χαμηλής πολυπλοκότητάς του (βλέπε κεφάλαιο 3) . Για τους αριθμούς εγγραφών 100.000 και 200.000, οι χρόνοι εκτέλεσης των αλγορίθμων αυξάνονται γραμμικά και με μικρό ρυθμό αύξησης συναρτήσει του αριθμού των χαρακτηριστικών ενώ για 300.000 και 400.000 εγγραφές ο ρυθμός αύξησης είναι αρκετά μεγαλύτερος.



Εικόνα 4.8: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του MakeDensity συναρτήσει του αριθμού των εγγραφών

Στο διάγραμμα της εικόνας 4.8, παρατηρούμε ότι ο ρυθμός αύξησης του χρόνου εκτέλεσης του αλγορίθμου για αριθμούς εγγραφών μεγαλύτερους των 200.000 μεγαλώνει καθώς με την αύξηση του αριθμού των εγγραφών.

4.2.4 Αποτελέσματα Αλγορίθμου Filterer

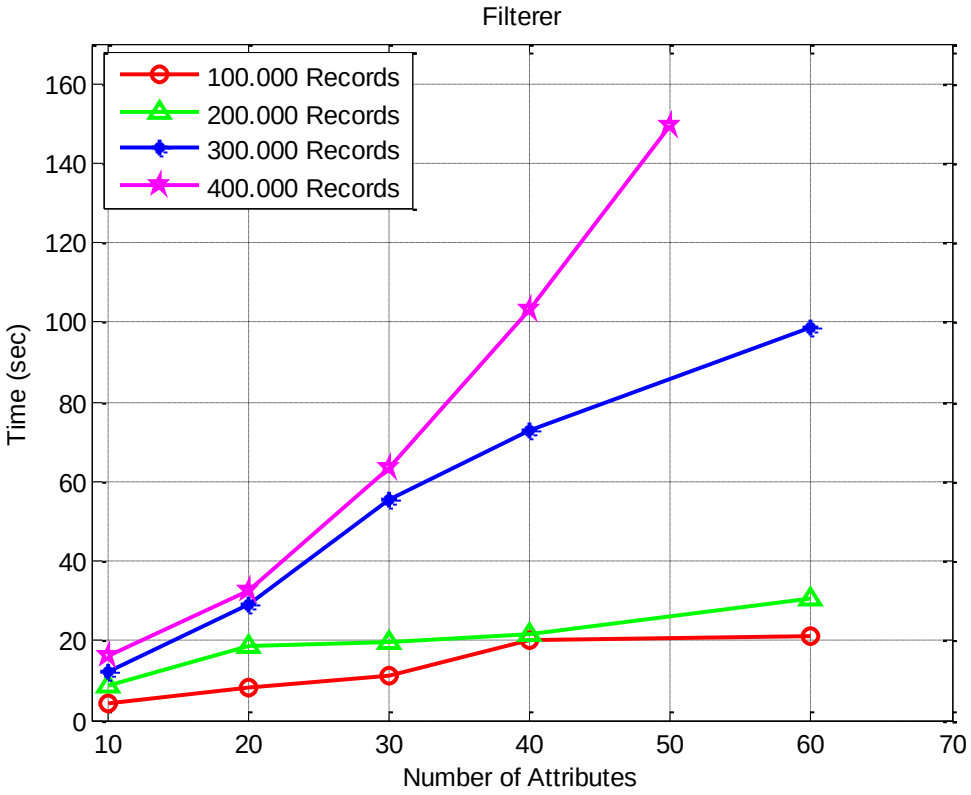
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- clusterer: SimpleKMeans (βλέπε τις παραμέτρους του SimpleKMeans)
- filter: AllFilter

Στον πίνακα 4.6 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου Filterer συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.9.

Πίνακας Αποτελεσμάτων αλγορίθμου Filterer - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	4.18	8.32	11.17	19.98	21.17
200.000	8.5	18.72	19.45	21.84	30.58
300.000	12.37	29	55.21	72.81	98.64
400.000	15.94	32.5	63.22	103.16	149.29

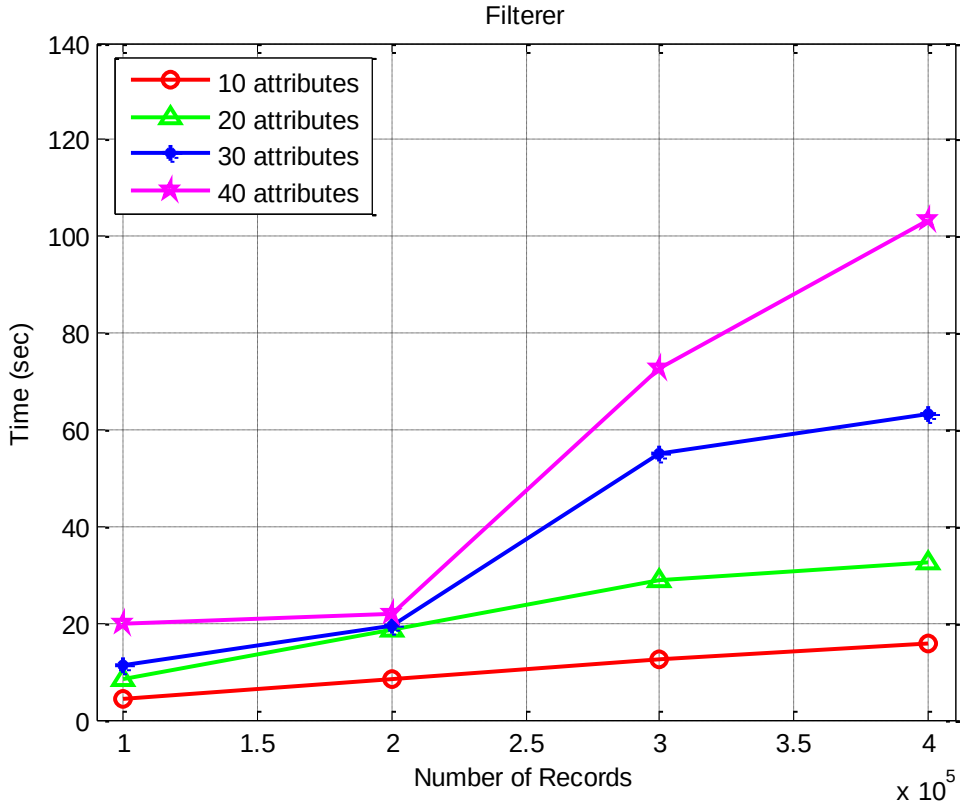
Πίνακας 4.6: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.9: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του Filterer συναρτήσει του αριθμού των χαρακτηριστικών

Όπως παρατηρούμε από το διάγραμμα της εικόνας 4.9, ο Filtered πετυχαίνει πολύ μικρούς χρόνους εκτέλεσης λόγω της χαμηλής πολυπλοκότητάς του (βλέπε κεφάλαιο 3) . Για τους αριθμούς εγγραφών 100.000 και 200.000, οι χρόνοι εκτέλεσης των αλγορίθμων αυξάνονται γραμμικά και με μικρό ρυθμό αύξησης συναρτήσει του αριθμού των χαρακτηριστικών ενώ για 300.000 και 400.000 εγγραφές, ο ρυθμός αύξησης είναι αρκετά μεγαλύτερος.

Μια σημαντική παρατήρηση για τις 300.000 και 400.000 είναι ότι για αριθμούς χαρακτηριστικών μέχρι 30, οι χρόνοι εκτέλεσής τους δεν παρουσιάζουν μεγάλη διαφορά. Καθώς ο αριθμός των χαρακτηριστικών είναι μεγαλύτερος από 30, παρουσιάζεται σημαντική αύξηση στους χρόνους εκτέλεσης για την περίπτωση των 400.000 εγγραφών.



Εικόνα 4.10: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του Filterer συναρτήσει του αριθμού των εγγραφών.

Στο διάγραμμα της εικόνας 4.10, παρατηρούμε ότι ο ρυθμός αύξησης του χρόνου εκτέλεσης του αλγορίθμου για αριθμούς εγγραφών μεγαλύτερους των 200.000, μεγαλώνει καθώς με την αύξηση του αριθμού των εγγραφών.

4.2.5 Αποτελέσματα Αλγορίθμου FarthestFirst

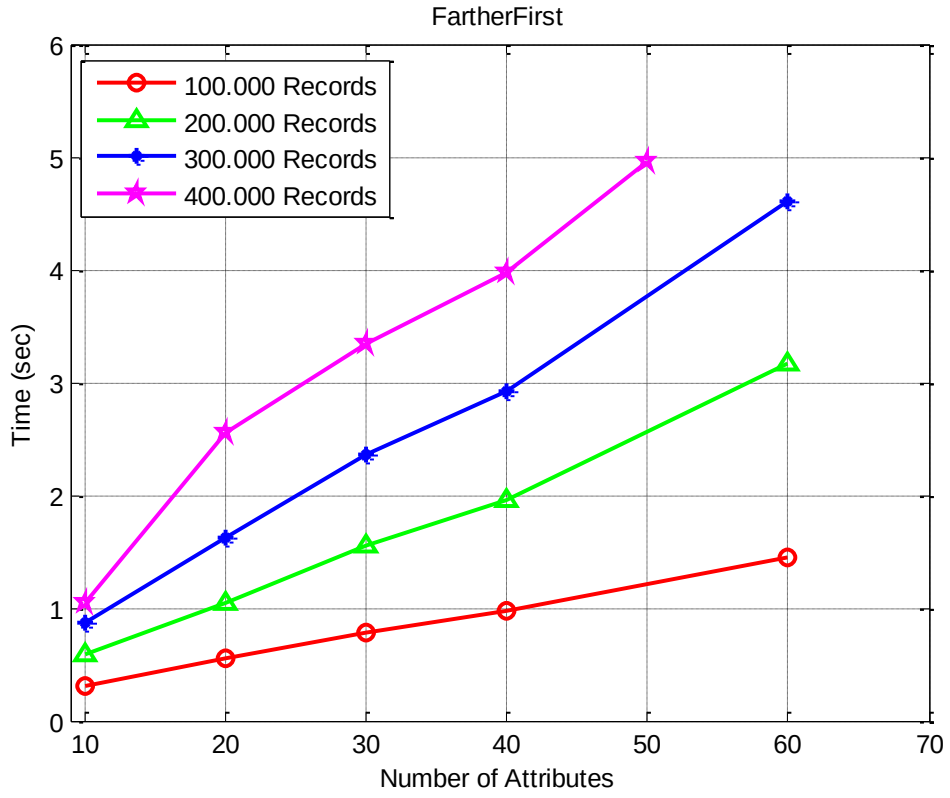
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- numClusters: 4
- seed: 1

Στον πίνακα 4.7 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου FarthestFirst συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.11.

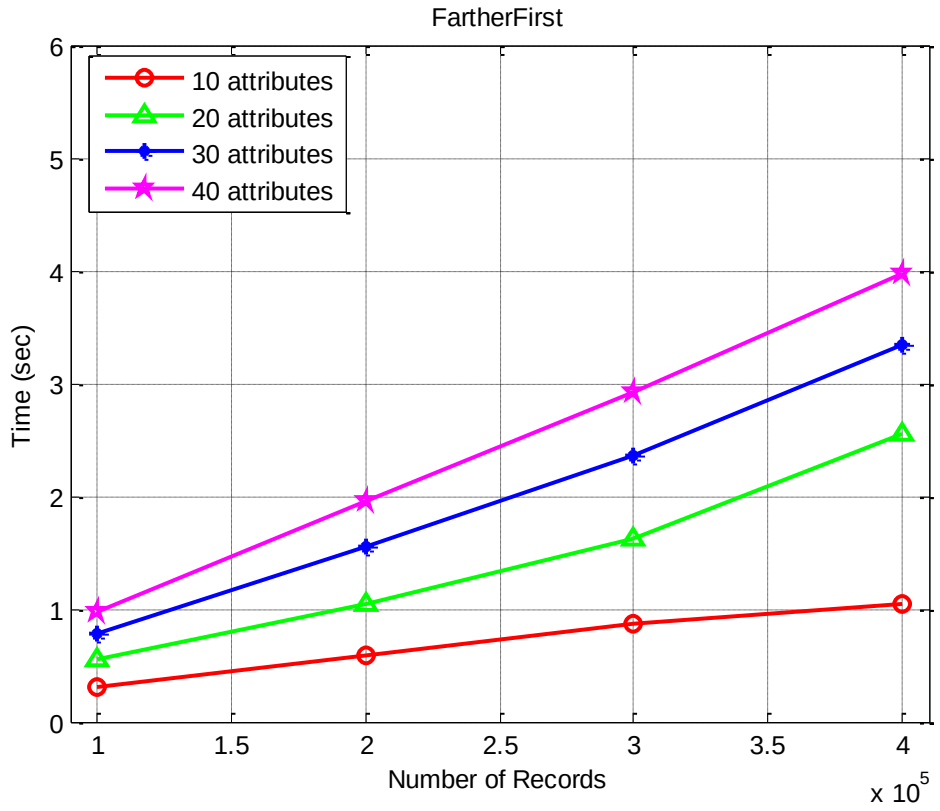
Πίνακας Αποτελεσμάτων αλγορίθμου FarthestFirst - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	0.3	0.56	0.78	0.97	1.45
200.000	0.58	1.05	1.56	1.95	3.17
300.000	0.86	1.62	2.36	2.92	4.61
400.000	1.04	2.55	3.35	3.97	4.96

Πίνακας 4.7: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.11: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του FarthestFirst συναρτήσει του αριθμού των χαρακτηριστικών.

Όπως παρατηρούμε από το διάγραμμα της εικόνας 4.11, ο FarthestFirst πετυχαίνει πολύ μικρούς χρόνους εκτέλεσης λόγω της χαμηλής πολυπλοκότητάς του (βλέπε κεφάλαιο 3). Επίσης οι ρυθμοί αύξησης για όλους τους αριθμούς των χαρακτηριστικών δεν παρουσιάζουν μεγάλες μεταβολές.



Εικόνα 4.12: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του FarthestFirst συναρτήσει του αριθμού των εγγραφών.

Στο διάγραμμα της εικόνας 4.12, παρατηρούμε ότι σε όλες τις περιπτώσεις αριθμού χαρακτηριστικών, οι χρόνοι εκτέλεσης του αλγορίθμου αυξάνουν γραμμικά συναρτήσει του αριθμού των εγγραφών.

Επίσης για τις περιπτώσεις του αριθμού χαρακτηριστικών 20, 30, 40 ο ρυθμός αύξησης του χρόνου εκτέλεσης, παρουσιάζει μικρές μεταβολές.

4.2.6 Αποτελέσματα Αλγορίθμου EM

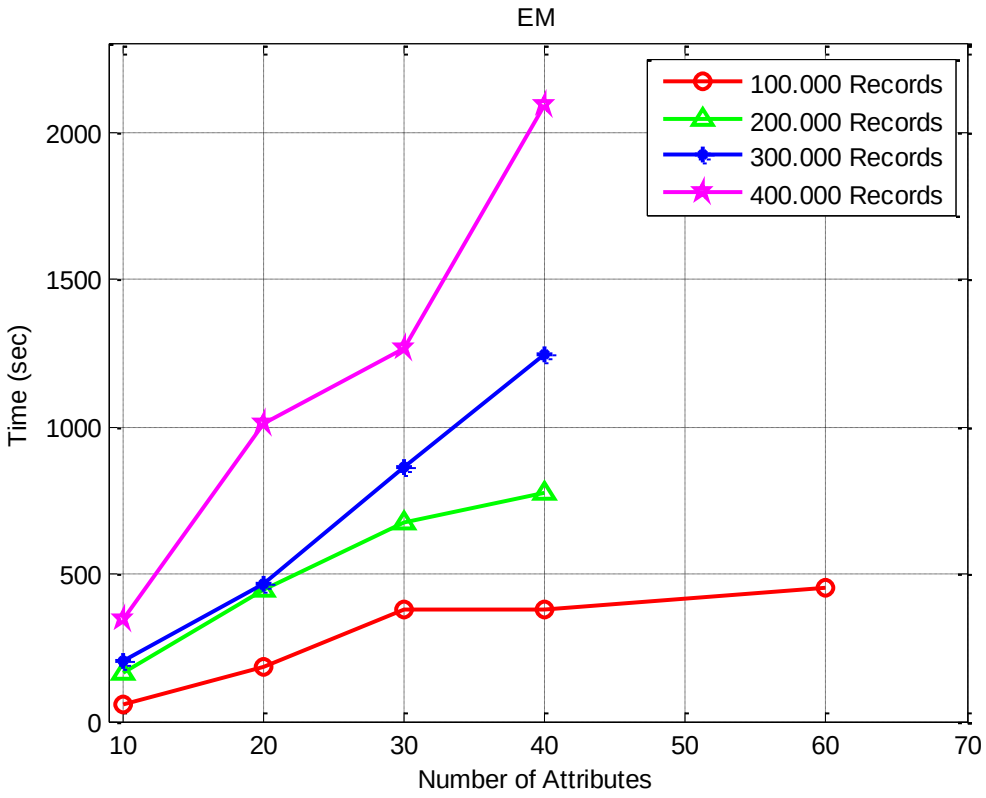
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- debug: False
- displayModelInOldFormat: False
- maxIterations: 100
- minStdDev: 1.0E-6
- numClusters: -1
- seed: 100

Στον πίνακα 4.7 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου EM συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.13.

Πίνακας Αποτελεσμάτων αλγορίθμου ΕΜ - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	57.78	187.02	383.09	378.46	451.33
200.000	164.64	448.72	679.21	776.33	Nan
300.000	207.39	469.32	864.98	1246.86	Nan
400.000	345.7	1011.96	1266.72	2096.24	Nan

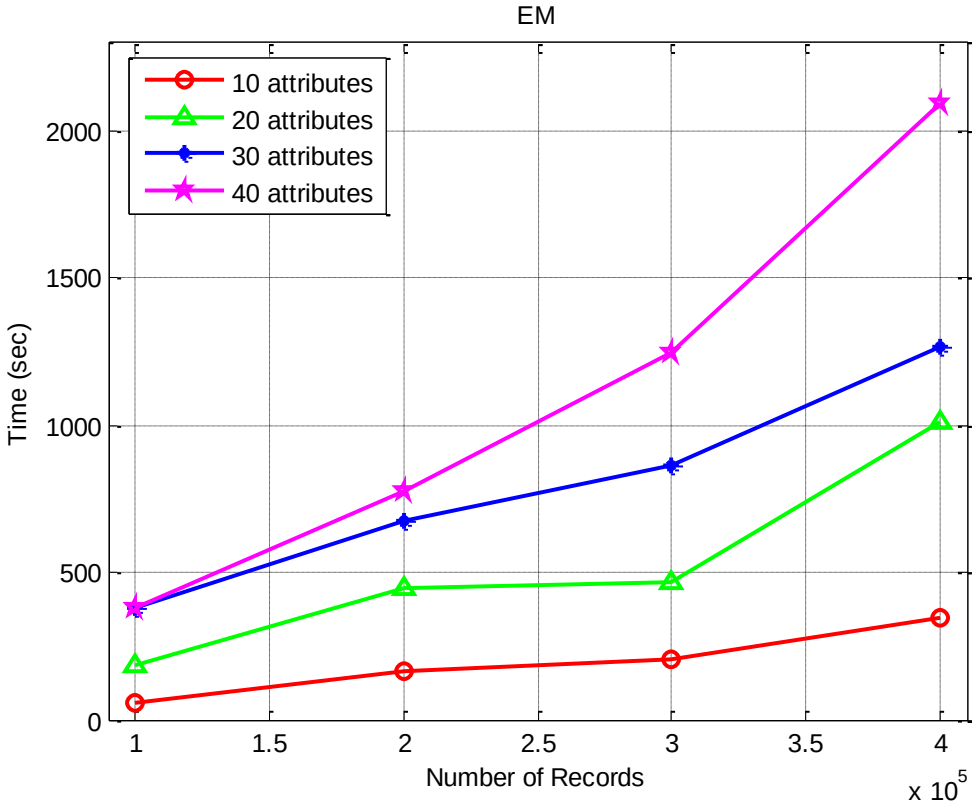
Πίνακας 4.8: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.13: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του ΕΜ συναρτήσει του αριθμού των χαρακτηριστικών.

Όπως παρατηρούμε από το διάγραμμα της εικόνας 4.13, ο ΕΜ λόγω της υψηλής του πολυπλοκότητάς (βλέπε κεφάλαιο 3) πετυχαίνει αυξημένους χρόνους εκτέλεσης σε σύγκριση με τους παραπάνω αλγορίθμους. Ο ρυθμός αύξησης του χρόνου εκτέλεσης αυξάνεται με την αύξηση του αριθμού των εγγραφών.

Τέλος για τους αριθμούς των εγγραφών 200.000, 300.000, 400.000, η εκτέλεση δεν ολοκληρώνεται όταν ο αριθμός χαρακτηριστικών ξεπεράσει τα 40.



Εικόνα 4.14: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του EM συναρτήσει του αριθμού των εγγραφών.

Στο διάγραμμα της εικόνας 4.14, παρατηρούμε ότι για κάθε τιμή χαρακτηριστικών την μεταβολή των χρόνων εκτέλεσης του αλγορίθμου συναρτήσει του αριθμού των εγγραφών. Όπως φαίνεται, όσο αυξάνεται ο αριθμός των χαρακτηριστικών και των εγγραφών, ο χρόνος εκτέλεσης του αλγορίθμου αυξάνει σημαντικά και ιδιαίτερα όταν έχουμε μεγάλες τιμές αριθμού εγγραφών.

4.2.7 Αποτελέσματα Αλγορίθμου DBScan

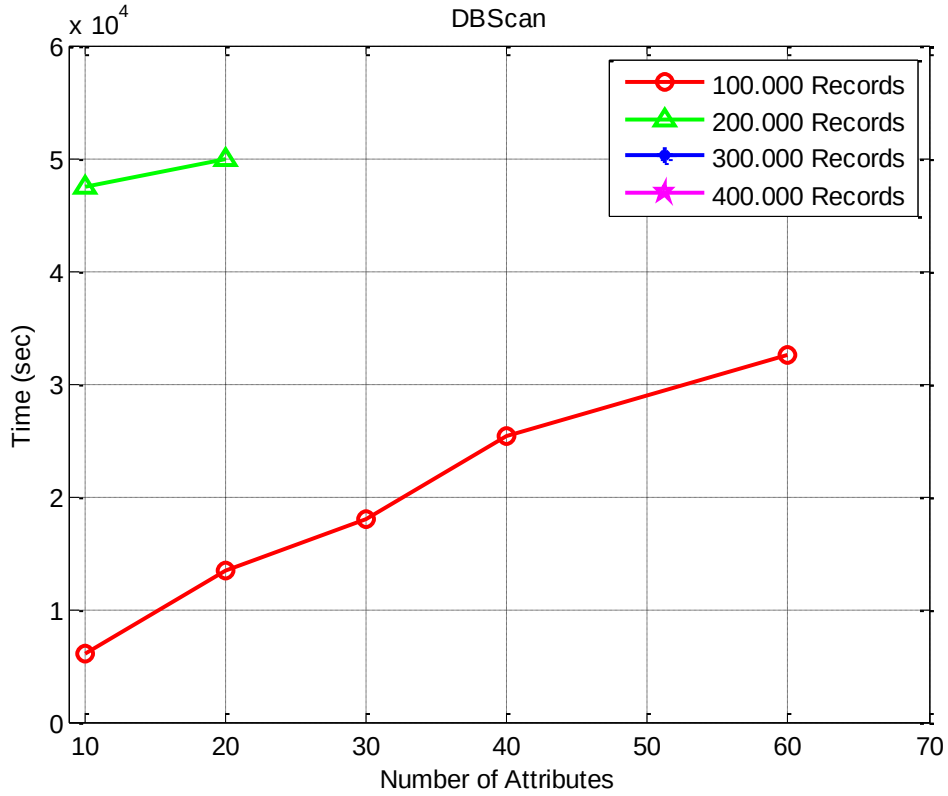
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- database_Type: WEKA.clusterers.forOPTICSAndDBScan.Databases.SequentialDatabase
- database_distanceType: WEKA.clusterers.forOPTICSAndDBScan.DataObjects.EuclideanDataObject
- epsilon: 0.9
- minPoints: 6

Στον πίνακα 4.9 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου DBScan συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.15.

Πίνακας Αποτελεσμάτων αλγορίθμου DBScan - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	6062.3	13508.32	17933.69	25436.99	32557.08
200.000	47474.01	49868.04	Nan	Nan	Nan
300.000	Nan	Nan	Nan	Nan	Nan
400.000	Nan	Nan	Nan	Nan	Nan

Πίνακας 4.9: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.15: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του DBScan συναρτήσει του αριθμού των χαρακτηριστικών.

Ο DBScan χαρακτηρίζεται από υψηλή πολυπλοκότητα, γεγονός που εξηγεί τους πολύ υψηλούς χρόνους εκτέλεσης. Όπως παρατηρούμε στο διάγραμμα της εικόνας 4.15, ο DBScan μπόρεσε να ολοκληρώσει την εκτέλεσή του για όλα τα χαρακτηριστικά στις περιπτώσεις των 100.000, καθώς και για 10 και 20 χαρακτηριστικά στην περίπτωση των 200.000 εγγραφών, ενώ ήταν αδύνατη η ολοκλήρωση της εκτέλεσής του για τις υπόλοιπες περιπτώσεις.

Επειδή σε περιπτώσεις μεγάλου αριθμού χαρακτηριστικών και εγγραφών, η εκτέλεση του αλγορίθμου DBScan δεν μπορεί να ολοκληρωθεί, η εξαγωγή συμπερασμάτων βάσει του διαγράμματος χρόνου εκτέλεσης συναρτήσει του αριθμού των εγγραφών ήταν αδύνατη. Το ίδιο ισχύει και για τους παρακάτω αλγορίθμους CobWeb και Optics.

4.2.8 Αποτελέσματα Αλγορίθμου CobWeb

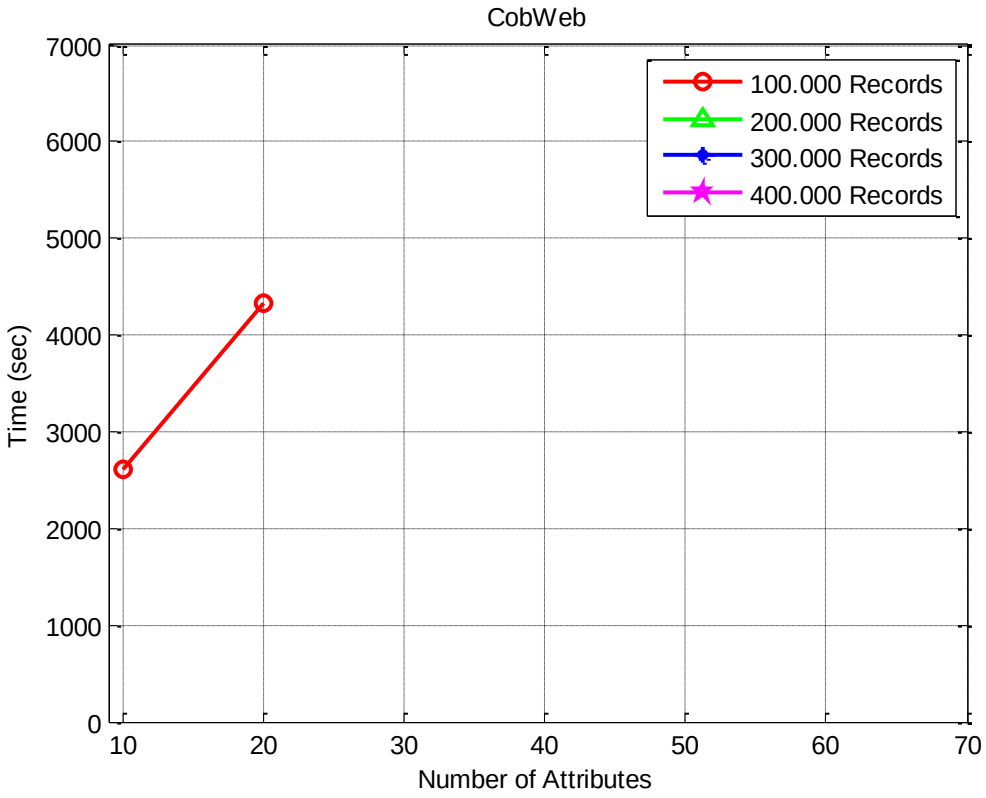
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- acuity: 1.0
- cutoff: 0.0028209479177387815
- saveInstanceData: False
- seed: 42

Στον πίνακα 4.10 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου CobWeb συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.16.

Πίνακας Αποτελεσμάτων αλγορίθμου CobWeb - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	2604.28	4338.37	Nan	Nan	Nan
200.000	Nan	Nan	Nan	Nan	Nan
300.000	Nan	Nan	Nan	Nan	Nan
400.000	Nan	Nan	Nan	Nan	Nan

Πίνακας 4.10: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.



Εικόνα 4.16: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του CobWeb συναρτήσει του αριθμού των χαρακτηριστικών.

Ο Cobweb χαρακτηρίζεται από υψηλή πολυπλοκότητα. Στο διάγραμμα της εικόνας 4.16, φαίνεται ότι ο CobWeb μπόρεσε να ολοκληρώσει την εκτέλεσή του για τα 10 και 20 χαρακτηριστικά

στην περίπτωση των 100.000 εγγραφών, ενώ η ολοκλήρωση της εκτέλεσής του με χρήση μεγαλύτερων αρχείων δεδομένων ήταν αδύνατη.

4.2.9 Αποτελέσματα Αλγορίθμου Optics

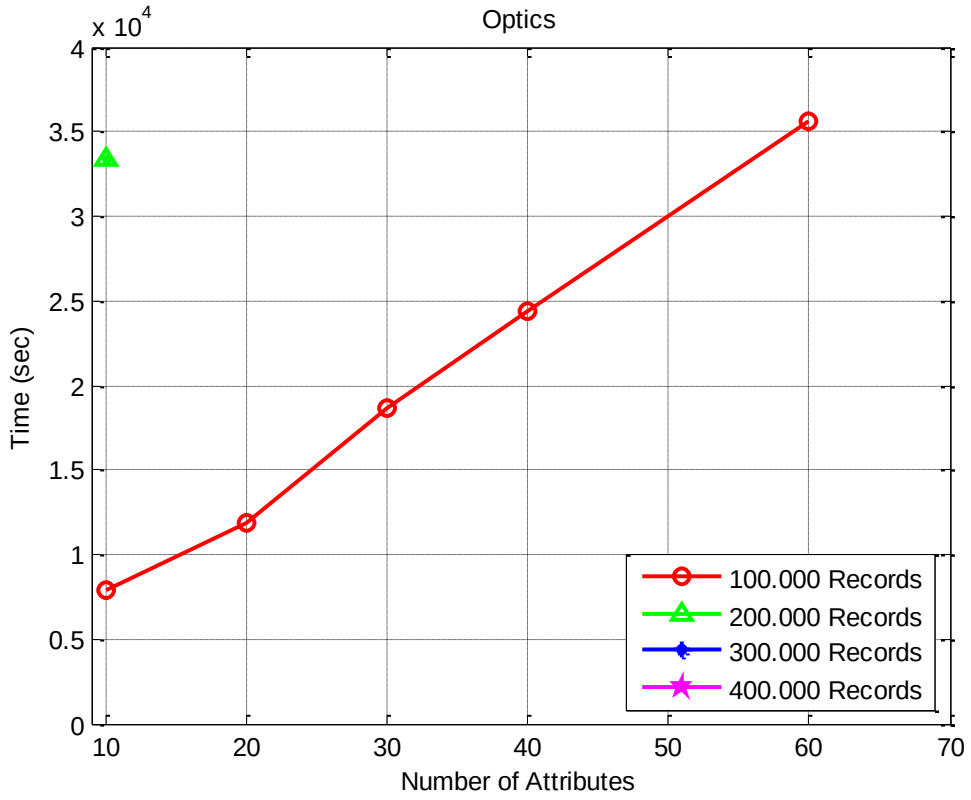
Για την εκτέλεση των πειραμάτων του αλγορίθμου στο WEKA, έγιναν οι παρακάτω αρχικοποιήσεις στις παραμέτρους του:

- databaseOutput: -
- database_Type: WEKA.clusterers.forOPTICSAndDBScan.Databases.SequentialDatabase
- database_distanceType: WEKA.clusterers.forOPTICSAndDBScan.DataObjects.EuclideanDataObject
- epsilon: 0.9
- minPoints: 6
- showGUI: True
- writeOPTICSresults: False

Στον πίνακα 4.11 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου Optics συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 4.17.

Πίνακας Αποτελεσμάτων αλγορίθμου Optics - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	7933.02	11909.84	18695.79	24331.22	35573.75
200.000	33369.37	Nan	Nan	Nan	Nan
300.000	Nan	Nan	Nan	Nan	Nan
400.000	Nan	Nan	Nan	Nan	Nan

Πίνακας 4.11: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.

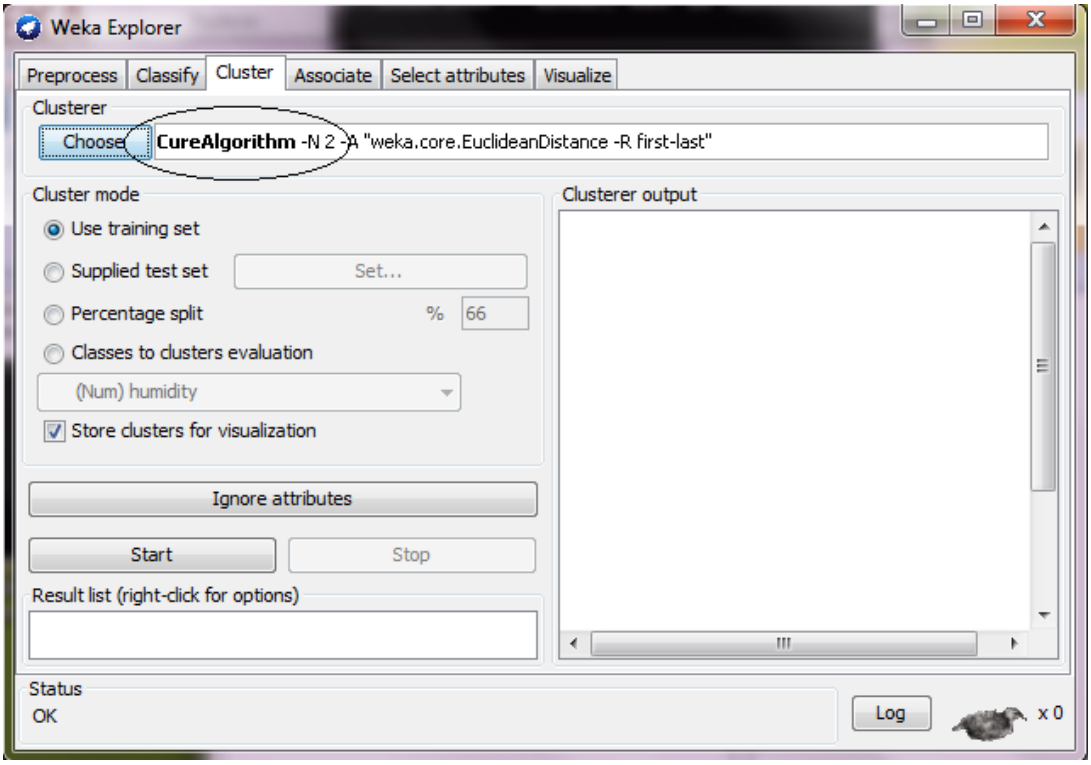


Εικόνα 4.17: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του Optics συναρτήσει του αριθμού των χαρακτηριστικών.

Όπως φαίνεται στο διάγραμμα της εικόνας 4.17, ο Optics μπόρεσε να ολοκληρώσει την εκτέλεσή του για όλα τα χαρακτηριστικά στην περίπτωση των 100.000 εγγραφών, καθώς και για την περίπτωση των 200.000 με 10 χαρακτηριστικά, ενώ η ολοκλήρωση της εκτέλεσής του με χρήση μεγαλύτερων αρχείων δεδομένων ήταν αδύνατη. Οι χρόνοι εκτέλεσής του είναι αρκετά υψηλοί, γεγονός που εξηγείται λόγω της υψηλής πολυπλοκότητάς του.

Κεφάλαιο 5 - Ο Αλγόριθμος CURE

Στο προηγούμενο κεφάλαιο παρουσιάσαμε τα αποτελέσματα των αλγορίθμων συσταδοποίησης που εμπεριέχονται στο σύστημα WEKA, και δείξαμε πως κλιμακώνονται με διαφορετικής διαστατικότητας αρχεία δεδομένων. Στο κεφάλαιο αυτό θα παρουσιάσουμε τον αλγόριθμο CURE (Clustering Using REpresentatives) ο οποίος μας δίνει την δυνατότητα να ξεπεράσουμε αρκετά μειονεκτήματα που παρουσιάζουν οι προηγούμενοι αλγόριθμοι όπως η ύπαρξη απομονωμένων σημείων (outliers), δυνατότητα συσταδοποίησης μεγαλύτερων αρχείων δεδομένων κτλ. Υλοποιήσαμε τον αλγόριθμο CURE όπως περιγράφεται στην εργασία των Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim [37] με χρήση της γλώσσας προγραμματισμού JAVA στην πλατφόρμα NetBeans και τον ενσωματώσαμε στο WEKA όπως φαίνεται και στην εικόνα 5.1.

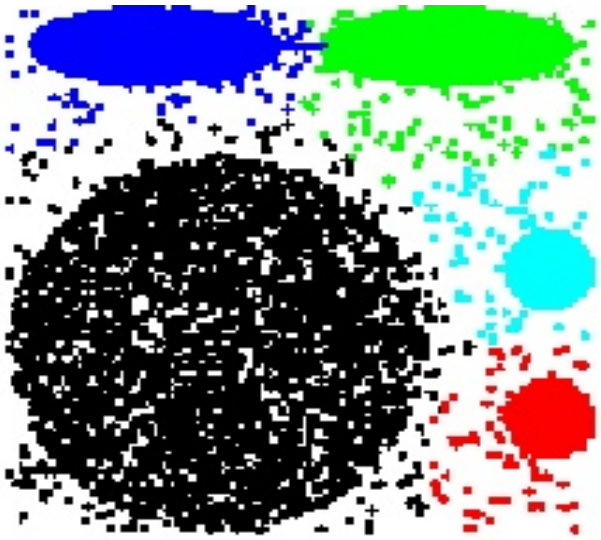


Εικόνα 5.1: Ενσωμάτωση του αλγορίθμου CURE στο σύστημα WEKA.

Τέλος θα παρουσιάσουμε αποτελέσματα που λάβαμε από την εκτέλεση του αλγορίθμου στο WEKA, τα οποία δείχνουν πως ο χρόνος εκτέλεσης του αλγορίθμου CURE κλιμακώνεται με τον αριθμό των χαρακτηριστικών και των εγγραφών ενός αρχείου δεδομένων καθώς και ποιοτικά αποτελέσματα που δείχνουν την δυνατότητα του αλγορίθμου να αναγνωρίζει σωστά, χωρικά ανομοιόμορφες συστάδες δεδομένων.

5.1 Παρουσίαση του αλγορίθμου CURE

Ο αλγόριθμος CURE είναι ένας αποδοτικός αλγόριθμος ο οποίος εφαρμόζεται σε μεγάλες βάσεις δεδομένων. Βασικό του χαρακτηριστικό είναι η ικανότητα του να αναγνωρίζει ομάδες οι οποίες έχουν μη σφαιρικά σχήματα (*πχ. ελλειψοειδή*) και η ευρωστία του σε περιπτώσεις που υπάρχουν ακραίες τιμές δεδομένων (*outliers*). Ο τρόπος με τον οποίο ο CURE επιλύει το πρόβλημα των συστάδων που δεν έχουν ομοιόμορφο μέγεθος ή σχήμα βασίζεται στην εφαρμογή ενός καινοτόμου συσσωρευτικού - ιεραρχικού (Agglomerative - Hierarchical) αλγορίθμου (βλέπε εικόνα 5.2).



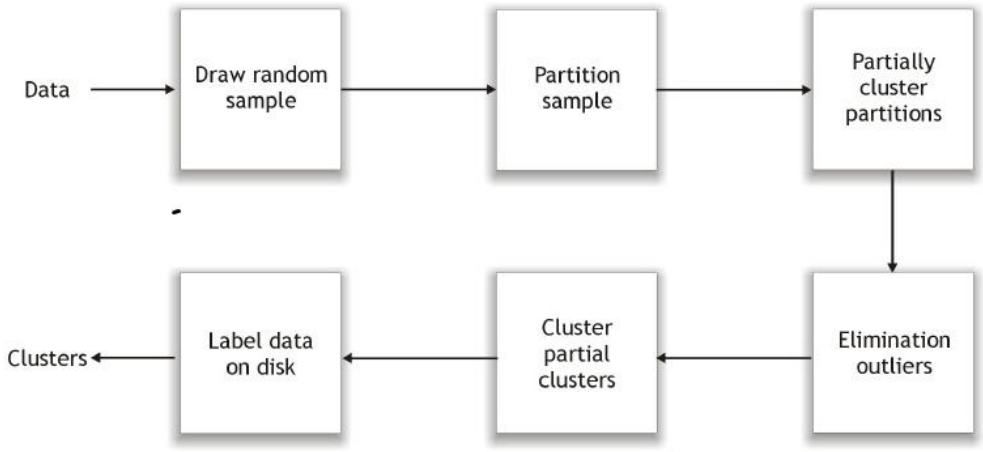
Εικόνα 5.2: Αποτέλεσμα συσταδοποίησης με χρήση του αλγορίθμου CURE. Παρατηρούμε ότι ο αλγόριθμος CURE έχει την δυνατότητα εξεύρεσης συστάδων με ανομοιόμορφα γεωμετρικά χαρακτηριστικά.

Ένα από τα βασικά χαρακτηριστικά του αλγορίθμου είναι η δυνατότητα του να χειρίζεται περιορισμένη μνήμη μιας και η συσταδοποίηση που πραγματοποιεί βασίζεται σε ένα τυχαίο δείγμα του συνόλου δεδομένων. Το τυχαίο δείγμα που επιλέγεται χωρίζεται σε έναν μικρότερο αριθμό δειγμάτων όπου κάθε ένα από αυτά συσταδοποιείται ξεχωριστά. Κάθε μία από τις συστάδες που προκύπτουν συσταδοποιούνται στην συνέχεια πλήρως, σε ένα δεύτερο πέρασμα. Αξίζει να σημειωθεί ότι η δειγματοληψία και ο διαχωρισμός των δεδομένων γίνονται μεμονωμένα, πράγμα που εξασφαλίζει ότι τα δεδομένα μπορούν να χωρέσουν στην διαθέσιμη μνήμη του συστήματος. Μετά την ολοκλήρωση της συσταδοποίησης του δείγματος, ο αλγόριθμος αναθέτει τα δεδομένα του συνολικού αρχείου στις συστάδες χρησιμοποιώντας την λογική ότι κάθε στοιχείο ανήκει στην συστάδα με τα πλησιέστερα αντιπροσωπευτικά σημεία (representatives).

5.2 Παράμετροι εισόδου του αλγορίθμου CURE

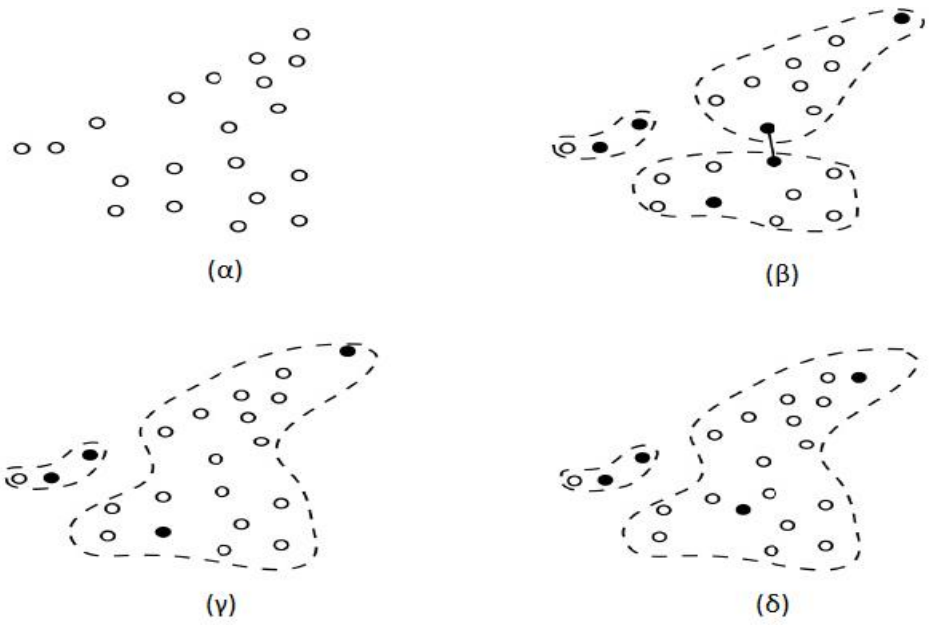
- Παράμετρος k: Δηλώνει τον αριθμό των συστάδων στις οποίες θα διαχωριστούν τα δεδομένα.
- Παράμετρος συρρίκνωσης α: Η παράμετρος αυτή λαμβάνει τιμές από 0 έως 1 και ο σκοπός της περιγράφεται στην συνέχεια του κεφαλαίου.
- Παράμετρος c: Ο αριθμός των αντιπροσωπευτικών σημείων (Representatives) κάθε συστάδας.
- Παράμετρος p: Ο αριθμός των τμημάτων που διαχωρίζεται το δείγμα των δεδομένων.

Παρακάτω παρουσιάζονται περιγραφικά τα βασικά βήματα του αλγορίθμου (βλέπε εικόνα 5.3, εικόνα 5.4):



Εικόνα 5.3: Περιγραφή αλγορίθμου CURE

Βήμα 1ο: Απομόνωση ενός δείγματος μεγέθους n από το αρχείο δεδομένων.
Βήμα 2ο: Διαχωρισμός του επιλεγμένου δείγματος σε p τεμάχια ίσου μεγέθους (κάθε τεμάχιο έχει μέγεθος n/p). Αυτός ο τεμαχισμός μας δίνει την δυνατότητα παραλληλοποίησης της συσταδοποίησης του δείγματος μιας και η συσταδοποίηση σε κάθε τεμάχιο μπορεί να εκτελεστεί ανεξάρτητα των υπολοίπων.
Βήμα 3ο: Χρησιμοποιώντας τον ιεραρχικό αλγόριθμο συσταδοποιούμε κάθε τεμάχιο και για κάθε ένα από αυτά βρίσκουμε ένα σύνολο από αντιπροσωπευτικά σημεία για κάθε συστάδα. Το πλήθος των συστάδων που δημιουργούνται είναι n/(p *q), όπου q μια σταθερά > 1. Για την εφαρμογή του ιεραρχικού αλγορίθμου μπορούν να χρησιμοποιηθούν διαφόρου τύπου μετρικές που υπολογίζουν τις αποστάσεις μεταξύ των σημείων όπως είναι η Ευκλείδεια, Manhattan κ.τ.λ.
Βήμα 4ο: Σε αυτό το βήμα πραγματοποιείται η απομάκρυνση των ακραίων σημείων (outliers). Για τον αλγόριθμο CURE έχουν προταθεί δύο βασικές τεχνικές απομάκρυνσης των ακραίων σημείων. Η πρώτη τεχνική αποβάλλει τις συστάδες που μεγαλώνουν με αργό ρυθμό. Για παράδειγμα οι συστάδες οι οποίες έχουν πολύ μικρό αριθμό στοιχείων σε σχέση με άλλες κατά την διάρκεια της συσταδοποίησης απορρίπτονται. Η δεύτερη τεχνική απομακρύνει τις πολύ μικρές συστάδες μετά το πέρας την διαδικασίας της συσταδοποίησης.
Βήμα 5ο: Σε αυτό το βήμα πραγματοποιείται η συσταδοποίηση όλων των δεδομένων του δείγματος και βρίσκονται οι τελικοί αντιπρόσωποι των τελικών συστάδων. Προκειμένου να γίνει αυτό πιο αποδοτικά, ο αλγόριθμος χρησιμοποιεί τους αντιπροσώπους των συστάδων που βρέθηκαν σε κάθε τμήμα κατά την πραγματοποίηση του βήματος 3.
Βήμα 6ο: Χρησιμοποιώντας τους τελικούς αντιπροσώπους της κάθε συστάδας (κάθε συστάδα αποτελείται από c αντιπροσώπους), ο αλγόριθμος αντιστοιχεί κάθε στοιχείο του αρχείου δεδομένου στη συστάδα με το πλησιέστερο αντιπροσωπευτικό σημείο.



Εικόνα 5.4: Περιγραφή αλγορίθμου CURE. (α) Δείγμα δεδομένων, (β) Οι τρεις συστάδες με τα αντιπροσωπευτικά τους σημεία, (γ) Συγχώνευση των συστάδων με τα πλησιέστερα σημεία, (δ) Συρρίκνωση των αντιπροσωπευτικών σημείων

Όπως αναφέραμε ο αλγόριθμος CURE χρησιμοποιεί c αντιπροσωπευτικά σημεία για να ορίσει κάθε συστάδα. Αυτοί οι αντιπρόσωποι επιλέγονται με τέτοιο τρόπο ώστε να είναι όσο το δυνατόν πιο απομακρυσμένοι μεταξύ τους. Σκοπός αυτού του τρόπου επιλογής τους είναι να μπορούν μέσω των θέσεών τους να σκιαγραφούν το φυσικό σχήμα και την γεωγραφία της κάθε συστάδας. Παρόλα αυτά η τεχνική επιλογής των αντιπροσωπευτικών σημείων, με την μεγαλύτερη απόσταση μεταξύ τους, παρουσιάζει το εξής πρόβλημα κατά την εκτέλεση της συσταδοποίησης: επιλέγοντας τα απομακρυσμένα σημεία υπάρχει πιθανότητα οι αντιπρόσωποι δύο διαφορετικών συστάδων να βρίσκονται πολύ κοντά μεταξύ τους με αποτέλεσμα να συγχωνευτούν οι δύο αυτές ομάδες, γεγονός που μπορεί να μας οδηγήσει σε εσφαλμένη συσταδοποίηση. Προκειμένου να αποφευχθεί αυτό το πρόβλημα, οι συγγραφείς της εργασίας [37] κάνουν χρήση μιας παραμέτρου α, η οποία παίρνει τιμές από 0 έως 1 και χρησιμοποιείται κατά την διαδικασία της συγχώνευσης (βήμα ιεραρχικού αλγορίθμου) με σκοπό να συρρικνώσει προς το κέντρο της κάθε συστάδας (κατά ένα λόγο α) τα αντιπροσωπευτικά σημεία ώστε να αυξηθεί η μεταξύ τους απόσταση. Όταν η παράμετρος α πάρει την τιμή 1, η θέση των αντιπροσώπων είναι το κέντρο της συστάδας, ενώ όταν πάρει την τιμή 0 τότε διατηρούν τις αρχικές τους θέσεις. Τα σημεία αυτά μετά τη συρρίκνωση, χρησιμοποιούνται για την αναπαράσταση της συστάδας. Οι συστάδες με τα κοντινότερα ζευγάρια αντιπροσώπων (*Representatives*) είναι οι συστάδες που συγχωνεύονται σε κάθε βήμα του ιεραρχικού αλγορίθμου που εφαρμόζει ο CURE. Αυτή η διαδικασία επιτρέπει στον CURE να ταυτοποιεί τις συστάδες σωστά και να τις κάνει λιγότερο ευαίσθητες σε ακραία σημεία (outliers). Ο αλγόριθμος CURE ολοκληρώνεται όταν ο αριθμός των συστάδων που θα δημιουργηθούν, γίνει ίσος με τον αριθμό των συστάδων k που έχουμε ορίσει.

Ο αλγόριθμος CURE επειδή ανήκει στην κατηγορία ιεραρχικών αλγορίθμων, παρουσιάζει υψηλή πολυπλοκότητα. Προκειμένου να μειωθεί αυτή η μεγάλη πολυπλοκότητα χρησιμοποιούνται δομές δεδομένων, όπως τα K-d trees [11] όπου μειώνει την πολυπλοκότητα σε $O(n^2 \log n)$. Επειδή αυτή η πολυπλοκότητα είναι αρκετά υψηλή για να εφαρμοστεί ο αλγόριθμος σε μεγάλα αρχεία δεδομένων, χρησιμοποιούμε τεχνικές όπως Τυχαία Δειγματοληψία (*Random Sampling*), Διαχωρισμός (*Partitioning*) και Labeling των δεδομένων στο δίσκο, όπου αποτελούν και τις βασικές καινοτομίες του αλγορίθμμου CURE.

5.3 Αποτελέσματα Αλγορίθμου CURE

Για την εκτέλεση των πειραμάτων του αλγορίθμου CURE στο WEKA, επιλέξαμε τα ίδια αρχεία που χρησιμοποιήθηκαν στα πειράματα των αλγορίθμων του WEKA (βλέπε κεφάλαιο 4). Η αρχικοποίηση των τιμών των παραμέτρων δίνεται παρακάτω:

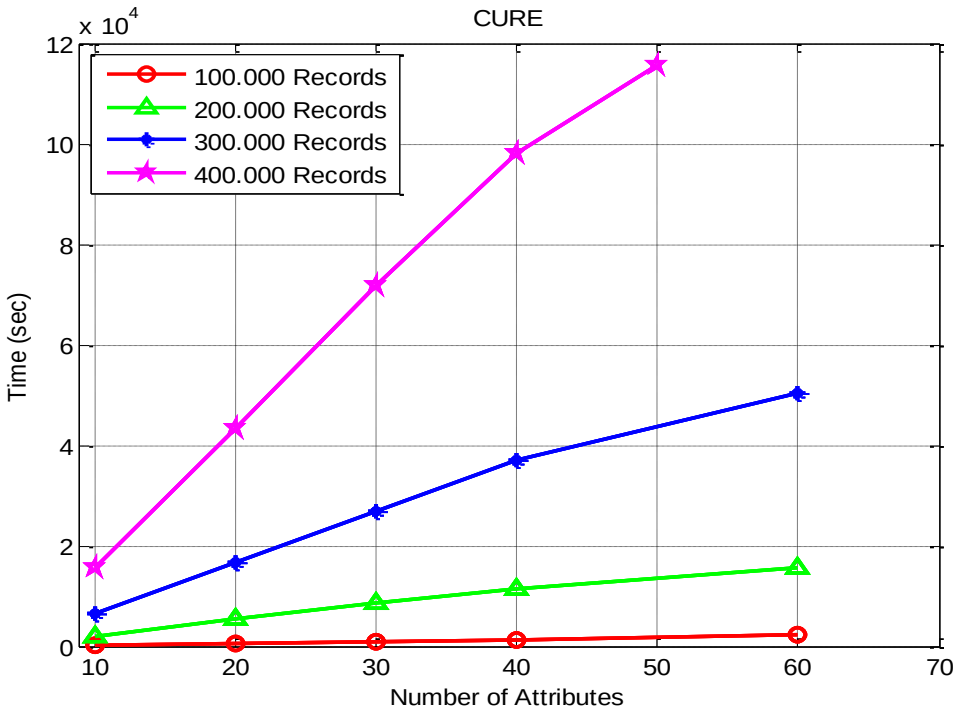
- Παράμετρος k: 4
- Παράμετρος συρρίκνωσης α: 0.7
- Παράμετρος c: 5
- Παράμετρος p: 20

Στον πίνακα 5.1 φαίνονται για κάθε αριθμό εγγραφών οι χρόνοι εκτέλεσης του αλγορίθμου CURE συναρτήσει του αριθμού των χαρακτηριστικών. Η γραφική απεικόνιση των τιμών φαίνεται στην εικόνα 5.5.

Πίνακας Αποτελεσμάτων αλγορίθμου CURE - Χρόνος Εκτέλεσης (sec)					
Records/Attributes	10	20	30	40	60 (50 για την περίπτωση των 400.000)
100.000	209.71	629.13	979.54	1327.94	2131.27
200.000	1977.17	5355.98	8440.66	11356	15673.11
300.000	6637.02	16585.18	27013.06	36877.34	50276.23
400.000	15642.49	43304.78	71830.46	98125.78	115583.48

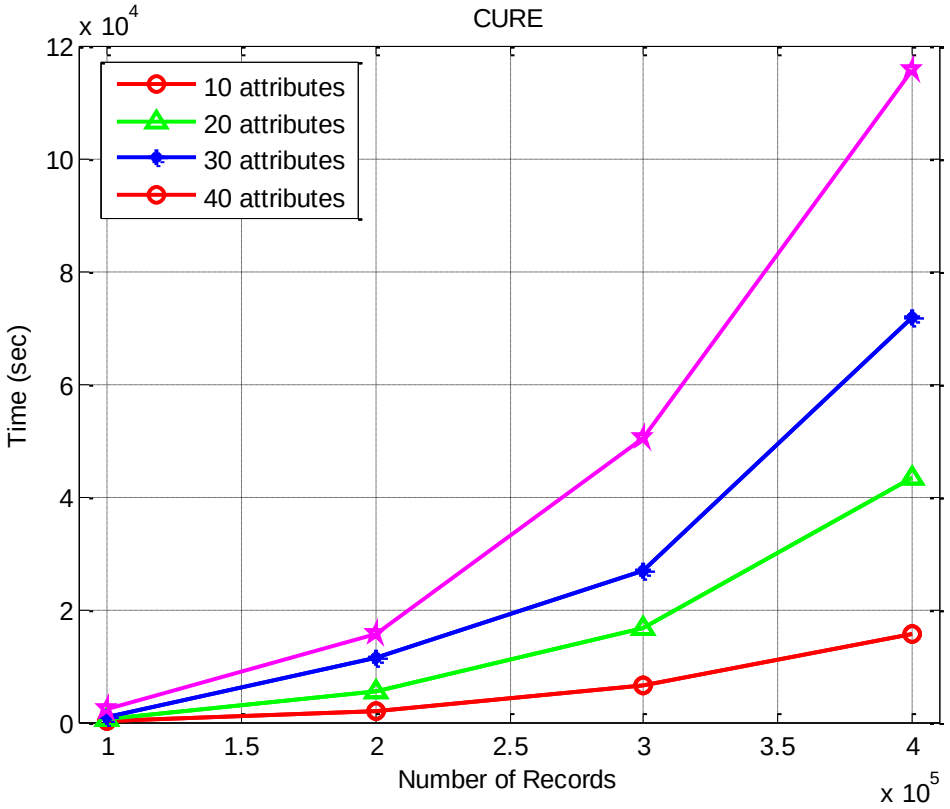
Πίνακας 5.1: Περιέχει τις τιμές των χρόνων εκτέλεσης για όλα τα σενάρια εκτέλεσης πειραμάτων με διαφορετικό αριθμό εγγραφών και χαρακτηριστικών.

Στα παρακάτω διαγράμματα γίνεται οπτικοποίηση των αποτελεσμάτων που παρουσιάζονται στον 5.1, με χρήση του πακέτου Matlab.



Εικόνα 5.5: Στο διάγραμμα φαίνονται για κάθε αριθμό εγγραφών, οι χρόνοι εκτέλεσης του CURE συναρτήσει του αριθμού των χαρακτηριστικών.

Όπως παρατηρούμε από το διάγραμμα της εικόνας 5.4, ο CURE περιγράφεται από πολύ μεγάλους χρόνους εκτέλεσης. Αυτό οφείλεται στην μεγάλη του πολυπλοκότητα. Ο χρόνος εκτέλεσης του αλγορίθμου αυξάνει με την αύξηση του αριθμού των χαρακτηριστικών και μάλιστα όσο μεγαλώνει ο αριθμός των εγγραφών, η αύξηση αυτή γίνεται όλο και πιο απότομη (οι αντίστοιχες ευθείες έχουν μεγαλύτερη κλίση).



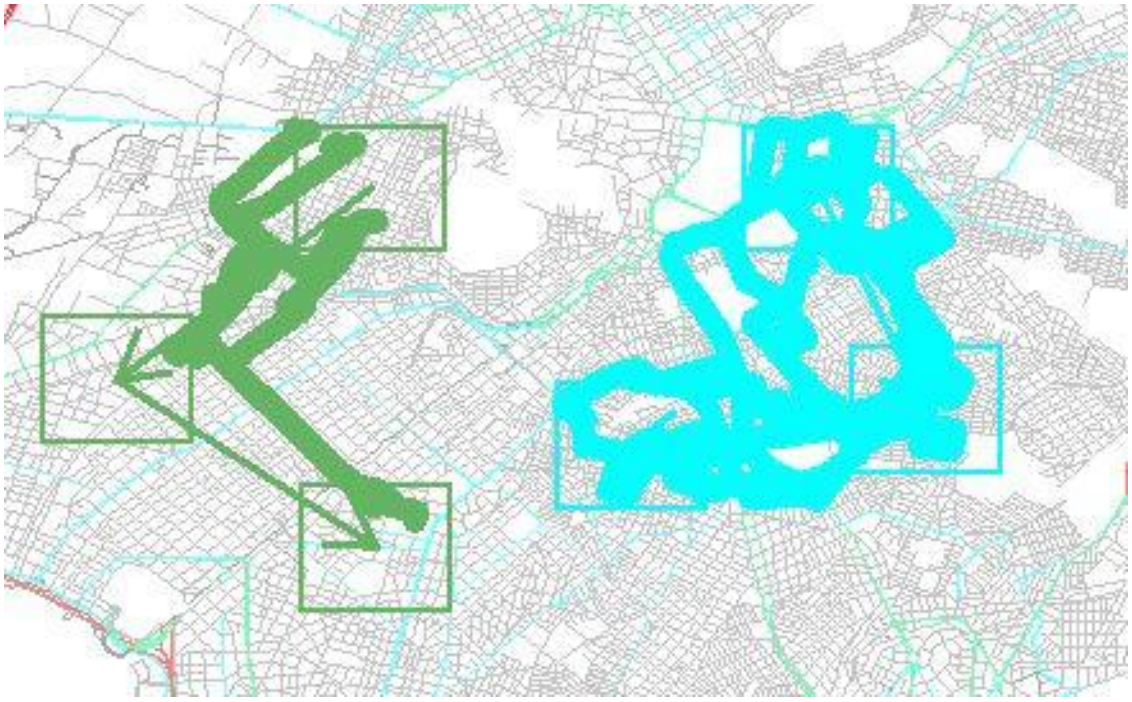
Εικόνα 5.6: Στο διάγραμμα φαίνονται για κάθε αριθμό χαρακτηριστικών οι χρόνοι εκτέλεσης του CURE συναρτήσει του αριθμού των εγγραφών.

Στο διάγραμμα της εικόνας 5.6, παρατηρούμε πώς ο χρόνος εκτέλεσης του αλγορίθμου, για κάθε αριθμό χαρακτηριστικών, αυξάνεται συναρτήσει του αριθμού των εγγραφών. Οι καμπύλες δείχνουν ότι ο χρόνος αυξάνεται μη γραμμικά με την αύξηση των εγγραφών, γεγονός που εξηγείται απόλυτα από την πολυπλοκότητα του αλγορίθμου. Τέλος αξίζει να σημειωθεί ότι όσο αυξάνεται ο αριθμός των χαρακτηριστικών, αυξάνεται και η μη γραμμικότητα των χρόνων εκτέλεσης συναρτήσει των εγγραφών.

Παρόλη την μεγάλη πολυπλοκότητα του αλγορίθμου, παρατηρούμε ότι η εκτέλεση του αλγορίθμου πραγματοποιήθηκε επιτυχώς για όλες τις περιπτώσεις, σε αντίθεση με κάποιους από τους αλγορίθμους που έχουν παρόμοια πολυπλοκότητα όπως DBScan και OPTICS. Ένας από τους βασικούς λόγους είναι ότι ο αλγόριθμος CURE επιλέγει ένα μικρό ποσοστό δεδομένων του συνολικού αρχείου για να πραγματοποιήσει την συσταδοποίηση (στην περίπτωση μας 5% του συνολικού αρχείου), γεγονός που του επιτρέπει να διαχειριστεί καλύτερα την διαθέσιμη μνήμη του συστήματος.

Για να επιβεβαιώσουμε την δυνατότητα του αλγορίθμου να αναγνωρίζει συστάδες δεδομένων οι οποίες είναι χωρικά ανομοιόμορφες χρησιμοποιήσαμε ένα αρχείο δεδομένων που έχει παραχθεί από το σύστημα Data Generator of Hermoupolis [38]. Η οπτικοποίηση του αρχείου φαίνεται στην εικόνα 5.7. Στην εικόνα παρατηρούμε δύο διαχωρίσιμες συστάδες δεδομένων

ανομοιόμορφου σχήματος τις οποίες θα προσπαθήσει ο αλγόριθμος CURE να αναγνωρίσει με επιτυχία. Αν και ο αριθμός των χαρακτηριστικών είναι μικρός (2 χαρακτηριστικά), το αρχείο περιέχει μεγάλο αριθμό εγγραφών (504.295).



Εικόνα 5.7: Οπτικοποίηση των δεδομένων του αρχείου δεδομένων από το σύστημα Data Generator of Hermoupolis

Για την εκτέλεση του πειράματος του αλγορίθμου CURE χρησιμοποιήθηκε το παραπάνω αρχείο. Η αρχικοποίηση των τιμών των παραμέτρων δίνεται παρακάτω:

- Παράμετρος k: 2
- Παράμετρος συρρίκνωσης α: 0.6
- Παράμετρος c: 5
- Παράμετρος p: 20

Χρησιμοποιώντας το groundtruth και τα αποτελέσματα συσταδοποίησης που προέκυψαν από τον αλγόριθμο CURE, διαμορφώσαμε τα ποιοτικά αποτελέσματα με την χρήση ενός confusion matrix που δίνεται στον πίνακα 5.2. Από τα αποτελέσματα του πίνακα παρατηρούμε ότι ο αλγόριθμος CURE αναγνώρισε σχεδόν με 100% επιτυχία τις πραγματικές συστάδες του αρχείου δεδομένων. Αξίζει να σημειωθεί ότι για την εκτέλεση του αλγορίθμου, το ποσοστό του αρχείου που χρησιμοποιήθηκε για την συσταδοποίηση ήταν το 5% των εγγραφών του συνολικού αρχείου. Έγιναν δοκιμές και με μεγαλύτερα ποσοστά αλλά τα αποτελέσματα παρέμειναν τα ίδια, γεγονός που δείχνει ότι ο αλγόριθμος CURE μπορεί να συσταδοποεί σωστά μεγάλα αρχεία δεδομένων χρησιμοποιώντας πολύ μικρό ποσοστό κάτι που τον κάνει πολύ αποδοτικό για εφαρμογές μεγάλων δεδομένων.

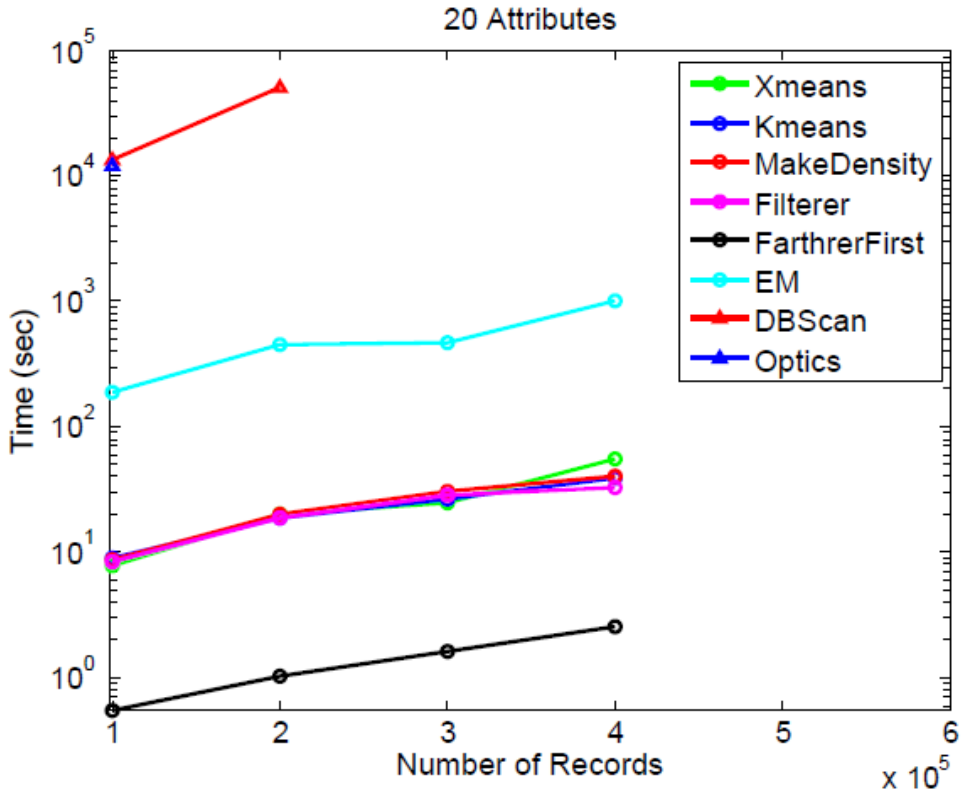
Confusion Matrix	**Πράσινο**	**Γαλάζιο**
Πράσινο	267.873	**3**
Γαλάζιο	1	236.418

Πίνακας 5.2: Ποιοτικά αποτελέσματα συσταδοποίησης του αλγορίθμου CURE. Όπως φαίνεται μόνο 3 (1) εγγραφές από τις 267.876 (236.419) κατηγοροιοποιήθηκαν λάθος από την πράσινη (μπλέ) συστάδα.

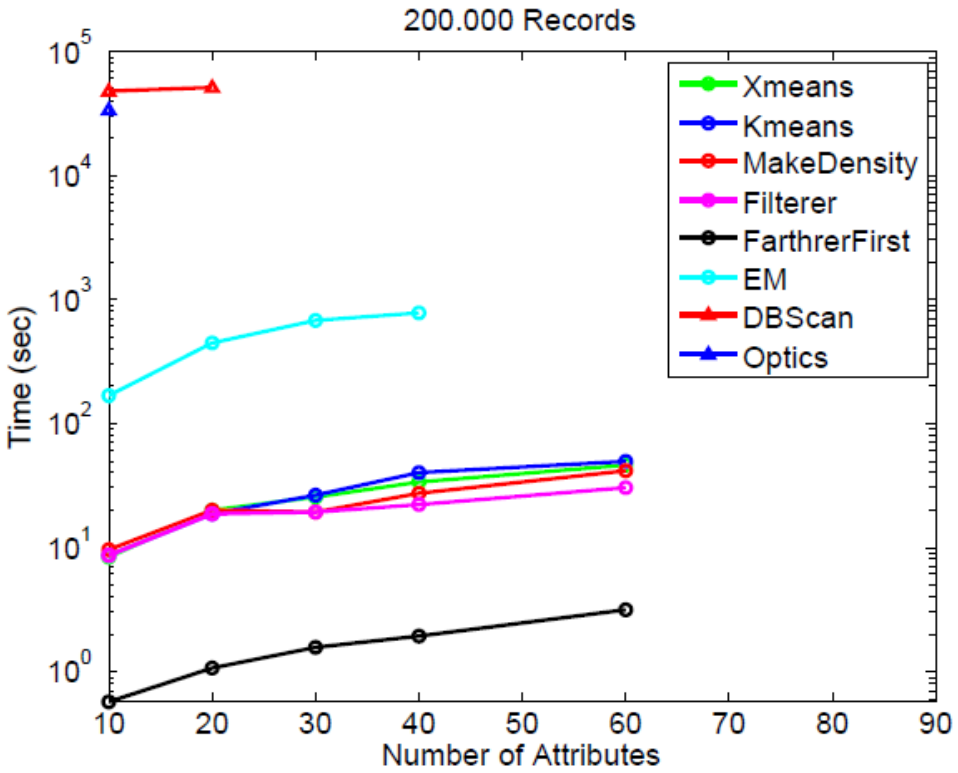
5.4 Σύγκριση Αλγορίθμων WEKA και Αλγορίθμου CURE

Στην ενότητα αυτή θα παρουσιάσουμε τα αποτελέσματα των αλγορίθμων συσταδοποίησης του WEKA και του αλγορίθμου CURE που υλοποιήσαμε και ενσωματώσαμε στο WEKA, με σκοπό την αποτύπωση των ορίων για το σύνολο των αλγορίθμων. Για την εκτέλεση των πειραμάτων χρησιμοποιήσαμε το αρχείο "USCensus1990" και δημιουργήσαμε αρχεία με αριθμό εγγραφών {100.000, 200.000, 300.000, 400.000} και με αριθμό χαρακτηριστικών 20 και αρχεία με αριθμό εγγραφών 200.000 και αριθμό χαρακτηριστικών {10, 20, 30, 40, 60}. Η επιλογή των εγγραφών και των χαρακτηριστικών έγινε με τυχαίο τρόπο.

Στις εικόνες 5.8 και 5.9 παρατηρούμε ότι βάσει του πίνακα πολυπλοκότητας (βλέπε Παράρτημα 1), οι αλγόριθμοι που ανήκουν στην κατηγορία των μη-ιεραρχικών ή αλλιώς διαμεριστικών αλγορίθμων συσταδοποίησης πετυχαίνουν πολύ μικρούς χρόνους εκτέλεσης λόγω της χαμηλής πολυπλοκότητάς τους. Αντιθέτως οι αλγόριθμοι που ανήκουν στην κατηγορία ιεραρχικών αλγορίθμων και εκείνοι που βασίζονται στην πυκνότητα του χώρου δεδομένων, χαρακτηρίζονται από πολύ υψηλή πολυπλοκότητα γεγονός που δικαιολογεί τους υψηλούς χρόνους εκτέλεσης.

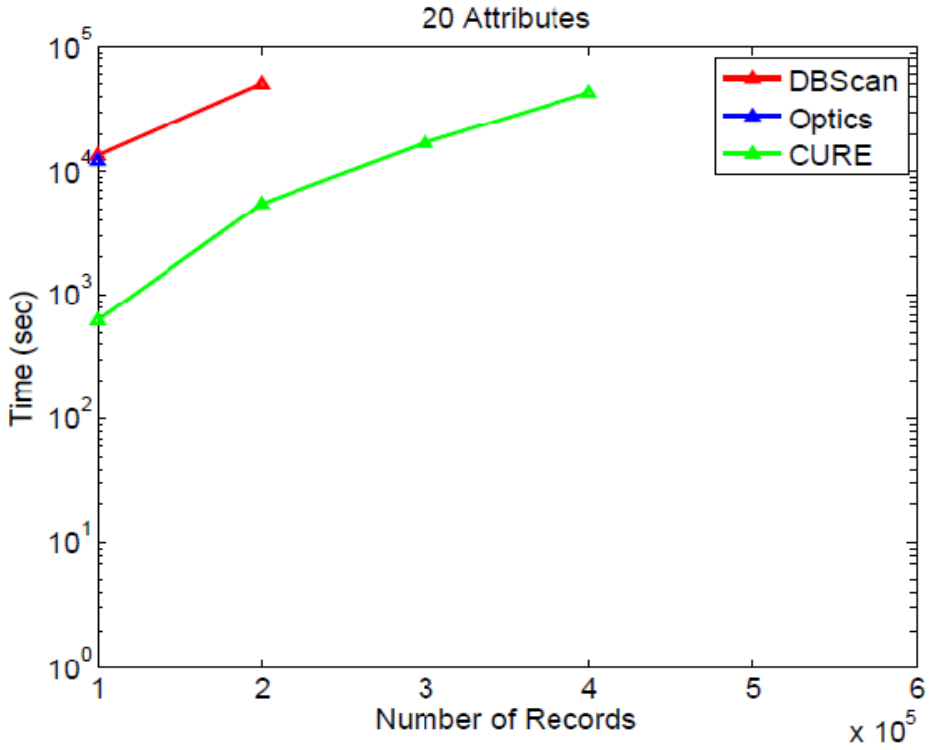


Εικόνα 5.8: Στο διάγραμμα φαίνονται σε λογαριθμική κλίμακα για 20 χαρακτηριστικά οι χρόνοι εκτέλεσης όλων των αλγορίθμων του WEKA συναρτήσει του αριθμού των εγγραφών.

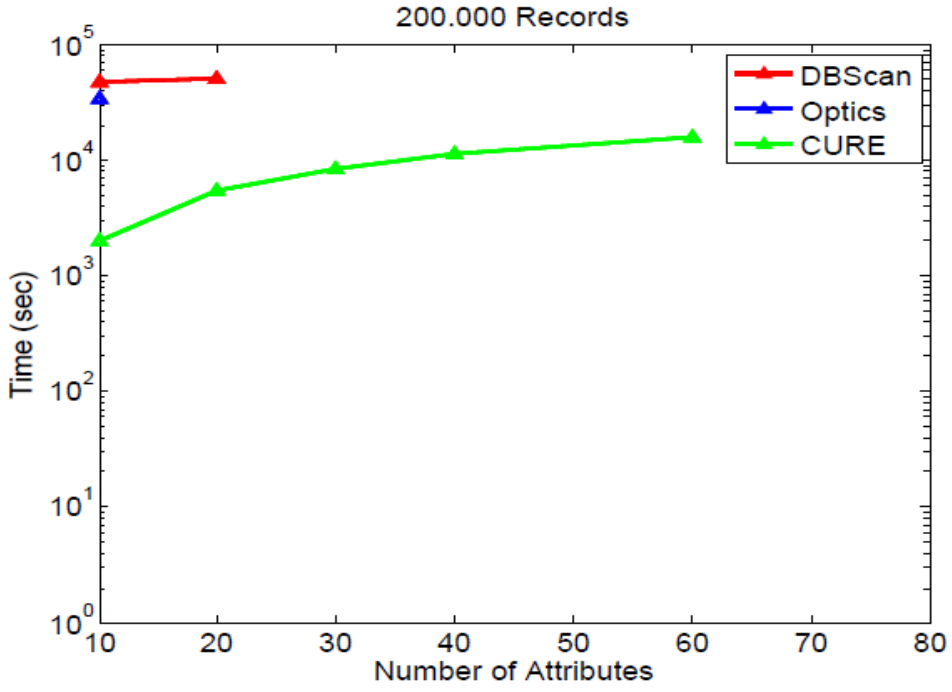


Εικόνα 5.9: Στο διάγραμμα φαίνονται σε λογαριθμική κλίμακα για αριθμό εγγραφών 200.000 οι χρόνοι εκτέλεσης όλων των αλγορίθμων του WEKA συναρτήσει του αριθμού των χαρακτηριστικών.

Στις εικόνες 5.10 και 5.11 φαίνονται οι χρόνοι εκτέλεσης των αλγορίθμων CURE, Optics και DBSCAN. Οι αλγόριθμοι αυτοί χαρακτηρίζονται από υψηλή πολυπλοκότητα και έχουν την δυνατότητα να αναγνωρίζουν χωρικά ανομοιόμορφες συστάδες δεδομένων. Από τα σχήματα παρατηρούμε ότι οι αλγόριθμοι DBSCAN και OPTICS πετυχαίνουν πολύ υψηλούς χρόνους εκτέλεσης και δεν καταφέρνουν να ολοκληρωθούν σε μεγαλύτερα αρχεία ενώ ο αλγόριθμος CURE πετυχαίνει καλύτερους χρόνους και μπορεί να εκτελεστεί σε πολύ μεγαλύτερα αρχεία δεδομένων. Ο βασικότερος λόγος αυτής της συμπεριφοράς οφείλεται στο γεγονός ότι ο αλγόριθμος CURE χρησιμοποιεί τεχνικές δειγματοληψίας και διαχωρισμού σε αντίθεση με τους αλγορίθμους DBSCAN και OPTICS.



Εικόνα 5.10: Στο διάγραμμα φαίνονται σε λογαριθμική κλίμακα για 20 χαρακτηριστικά, οι χρόνοι εκτέλεσης των αλγορίθμων CURE, Optics και DBSCAN συναρτήσει του αριθμού των εγγραφών.



Εικόνα 5.11: Στο διάγραμμα φαίνονται σε λογαριθμική κλίμακα για αριθμό εγγραφών 200.000, οι χρόνοι εκτέλεσης των αλγορίθμων CURE, Optics και DBSCAN συναρτήσει του αριθμού των χαρακτηριστικών.

Κεφάλαιο 6 - Συμπεράσματα / Μελλοντικές επεκτάσεις

Η Εξόρυξης Γνώσης είναι ένας ραγδαία αναπτυσσόμενος τομέας ο οποίος έχει επιρροή σε ένα μεγάλο αριθμό επιστημονικών κλάδων (πληροφορικής, βιολογίας, οικονομίας κ.τ.λ.). Οι βασικότερες τεχνικές είναι εμπνευσμένες από τους τομείς της μηχανικής μάθησης και της στατιστικής οι οποίοι είναι πολλά υποσχόμενοι για την εξέλιξη διαχείρισης και εξόρυξης από μεγάλες βάσεις δεδομένων. Επειδή η πολυπλοκότητα υλοποίησης των αλγορίθμων που χρησιμοποιούνται για την εξόρυξη γνώσης είναι μεγάλη, έχουν αναπτυχθεί λογισμικά τα οποία βοηθούν τον μέσο χρήστη να χρησιμοποιήσει τους αλγορίθμους αυτούς για ανάλυση των δεδομένων του για την ανάκτηση χρήσιμης πληροφορίας εξ' αυτών.

Στην παρούσα διπλωματική εργασία έγινε παρουσίαση του δημοφιλούς λογισμικού WEKA καθώς και των αλγορίθμων συσταδοποίησης που περιέχονται σε αυτό, με βασικό στόχο να αναδειχθούν τα πλεονεκτήματα και τα μειονεκτήματα καθενός από αυτούς. Επίσης έγινε αποτίμηση της ικανότητας των αλγορίθμων αυτών να διαχειριστούν μεγάλα αρχεία δεδομένων τα οποία περιείχαν μεγάλο αριθμό εγγραφών αλλά και χαρακτηριστικών. Μέσω μιας σειράς πειραμάτων έγινε αποτίμηση των αλγορίθμων ως προς το μέγιστο μέγεθος αρχείων που μπορούν να διαχειριστούν καθώς και του χρόνου εκτέλεσής τους συναρτήσει του αριθμού χαρακτηριστικών αλλά και των εγγραφών. Για κάθε αλγόριθμο παρουσιάσαμε και συγκρίναμε τα αποτελέσματα των πειραμάτων αυτών.

Υλοποιήσαμε σε γλώσσα προγραμματισμού JAVA τον δημοφιλή αλγόριθμο CURE ο οποίος θεωρείται ένας από τους καταλληλότερους αλγορίθμους για εξόρυξη γνώσης από μεγάλες βάσεις δεδομένων λόγω των πλεονεκτημάτων του που παρουσιάζουμε αναλυτικά στο κεφάλαιο 5. Προκειμένου να επαληθεύσουμε την λειτουργικότητα και αποδοτικότητα του εν λόγω αλγορίθμου όταν εκτελείται σε μεγάλα αρχεία δεδομένων με ανομοιόμορφες συστάδες, εκτελέστηκαν κατάλληλα πειράματα των οποίων τα αποτελέσματα παρουσιάζονται στο κεφάλαιο 5. Από τα πειράματα αυτά επιβεβαιώθηκε ότι ο αλγόριθμος CURE είναι σε θέση να διαχειριστεί αρχεία δεδομένων μεγάλης διαστατικότητας αλλά και να δώσει σωστά αποτελέσματα συσταδοποίησης.

Θα κλείσουμε την παρούσα διπλωματικής εργασία παρουσιάζοντας κάποιες κατευθύνσεις έρευνας οι οποίες θα είχαν ενδιαφέρον να μελετηθούν. Από τα αποτελέσματά μας έγινε σαφές ότι ο χρόνος εκτέλεσης των ιεραρχικών κυρίως αλγορίθμων είναι ιδιαίτερα μεγάλος όταν εκτελούνται σε αρχεία δεδομένων μεγάλης διαστατικότητας. Ο μεγάλος χρόνος αυτός εν μέρη οφείλεται στο γεγονός ότι η υλοποίηση των αλγορίθμων έγινε σε γλώσσα προγραμματισμού JAVA η οποία είναι ιδιαιτέρως αργή. Θα είχε ενδιαφέρον να μελετηθεί η βελτίωση των χρόνων εκτέλεσης των αλγορίθμων σε υλοποιήσεις γλωσσών προγραμματισμού χαμηλότερου επιπέδου ή ακόμα και σε hardware (FPGA) ή σε επεξεργαστές ειδικού σκοπού. Μία ενδιαφέρουσα επέκταση της παρούσας διπλωματικής είναι η μελέτη επιτάχυνσης των αλγορίθμων και ιδιαιτέρως του αλγορίθμου CURE μέσω παραλληλοποίησης τους. Θεωρούμε ότι η παραλληλοποίηση συγκεκριμένων βημάτων των αλγορίθμων θα μπορέσει να βελτιώσει σημαντικά τους χρόνους εκτέλεσής τους.

Τέλος αξίζει να μελετηθεί η απόδοση των αλγορίθμων ως προς την ποιότητα συσταδοποίησης χρησιμοποιώντας μεγάλα αρχεία δεδομένων, των οποίων είναι γνωστή εκ των προτέρων, η κατηγοριοποίηση των δεδομένων.

Παράρτημα 1: Πίνακας με την πολυπλοκότητα των αλγορίθμων

ΣΗΜΑΝΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΟΜΑΔΟΠΟΙΗΣΗΣ			
Όνομα	Πλεονεκτήματα	Μειονεκτήματα	Πολυπλοκότητα
SimpleKMeans	Απλότητα υλοποίησης, γραμμική πολυπλοκότητα	Το αποτέλεσμα βάσει αρχικής επιλογής κεντροειδών, απαιτείται η γνώση του πλήθους των ομάδων, το αποτέλεσμα επηρεάζεται αρνητικά σε μη σφαιρικά σχήματα, με την ύπαρξη ακραίων σημείων, και λόγω διαφορετικής πυκνότητας	O(n * K * I * d) n = number of points, K = number of clusters, I = number of iterations, d = number of attributes
Hierarchical	Τερματίζονται εύκολα, δεν απαιτείται η γνώση του πλήθους των ομάδων, ευκολία χειρισμού για κάθε μετρική	Υψηλές απαιτήσεις σε υπολογιστική ισχύ, δεν επιτρέπουν βελτίωση σε ήδη κατασκευασμένες συστάδες	O (n2)
CobWeb	Δεν απαιτείται η γνώση του πλήθους των ομάδων	Υψηλή πολυπλοκότητα	O(d n b2 log k) B=branching factor
DBSCAN	Δεν απαιτείται η γνώση του πλήθους των ομάδων, δημιουργία ομάδων σε ακανόνιστα σχήματα, ανίχνευση ακραίων σημείων	Απαιτείται η ακτίνα της ομάδας και ο ελάχιστος αριθμός σημείων	O(n^2)
EM	Καλά αποτελέσματα σε αρχεία δεδομένων με ελλιπή δεδομένα	Υψηλή πολυπλοκότητα	O(n * K * I * d)
FarthestFirst	Γρήγορος και κατάλληλος για δεδομένα μεγάλου όγκου	Απαιτείται η γνώση του πλήθους των ομάδων	O (n k)
X-Means	Δεν απαιτείται η γνώση του πλήθους των ομάδων	Χαμηλή πολυπλοκότητα	O (n)
OPTICS	Δεδομένα μεταβλητής πυκνότητας	Απαιτείται η ακτίνα της ομάδας και ο ελάχιστος αριθμός αντικειμένων	O (n log n)
CURE	Δημιουργία ομάδων σε μη σφαιρικά σχήματα, ανίχνευση outliers	Λόγω Δειγματοληψίας δεν χειρίζεται αποδοτικά κατηγορικά δεδομένα, απαιτείται η γνώση του πλήθους των ομάδων και των μελών αντιπροσώπων	O (n2 log n)

Παράρτημα 2: Κώδικας αλγορίθμου CURE σε JAVA

```
/*
* Algorithm CURE : find clusters from data file (sample) given in .arff type and then after finding representatives from the sample file
* , finds clusters from the original file (.arff) and then create a new file "CureCategoryFile.arff" having cluster as a new characteristic
*/
package javaapplication7;

import java.awt.Point;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;
import java.util.StringTokenizer;
import java.util.Vector;
import javax.swing.JOptionPane;
import WEKA.core.Attribute;
import WEKA.core.DistanceFunction;
import WEKA.core.EuclideanDistance;
import WEKA.core.Instance;
import WEKA.core.Instances;
import WEKA.core.converters.ArffSaver;
import WEKA.core.converters.ConverterUtils.DataSource;
/**
 *
 * @author Petty
 */

public class JavaApplication7 {

        private ArrayList<clustering_results> clustering_results = new ArrayList<clustering_results>();

        private List<TotalCl> Totalrepresentatives = new ArrayList<TotalCl>();

        private List<TotalCl> TotalMovedrepresentatives = new ArrayList<TotalCl>();
```

```
private List<Integer> numofCreatedClusters = new ArrayList<Integer>();

private  List<TotalCl> AllInstances = new ArrayList<TotalCl>();

private Integer representativesPercent = 5;

private Instance findCentroid;

private Integer m_NumClusters = 4;

private double alpha = 1.0;

private Instances m_Cluster;

/** the distance function used. */
//protected DistanceFunction m_DistanceFunction = new EuclideanDistance();

public void  buildClusterer(Instances data) throws Exception {

        Instances instances = new Instances(data);
        instances.setClassIndex(-1);

        for (int i = 0; i < instances.numInstances(); i++) {
                AllInstances.add(new TotalCl(instances.instance(i) , i));
        }

        int cl= 0;
        int counter = 0;

        List<TotalCl> Partition = new ArrayList<TotalCl>();

        int p= 20;  //partitions of file
        int q = 3 ; //(>1)
        int partition_count = AllInstances.size()/p;
        int part_count = 0;

         List<TotalCl> TotalPartitions_Movedrepresentatives = new ArrayList<TotalCl>();
         TotalPartitions_Movedrepresentatives.clear();

        for (int cc = 0; cc < p; cc++) {
                //fill tbl per partition
                List<TotalCl> PartitionTotalrepresentatives = new ArrayList<TotalCl>();
                List<TotalCl>  PartitionTotalMovedrepresentatives  =  new
ArrayList<TotalCl>();
                List<Integer> PartitionnumofCreatedClusters = new ArrayList<Integer>();
```

```
List<TotalCl> PartitionAllInstances = new ArrayList<TotalCl>();


for (int i = 0; i < partition_count ; i++) {
    //find number of clusters (get unique value from array list)
    PartitionnumofCreatedClusters.add(i+part_count);
    PartitionAllInstances.add(new
TotalCl(instances.instance(i+part_count) , i+part_count));
    PartitionTotalrepresentatives.add(new
TotalCl(instances.instance(i+part_count) , i+part_count));
    PartitionTotalMovedrepresentatives.add(new
TotalCl(instances.instance(i+part_count) , i+part_count));
}

int n = PartitionAllInstances.size();
int pk =  n / (p * q);  //clusters for every partition

Set<Integer> UniqueCreatedClusters1 = new
HashSet<Integer>(PartitionnumofCreatedClusters);

part_count += partition_count ;

while( UniqueCreatedClusters1.size() > pk ) {
    CalcMinDistance(PartitionTotalMovedrepresentatives.get(0),
PartitionAllInstances, PartitionTotalrepresentatives, PartitionTotalMovedrepresentatives, counter );

    counter +=1;

    UniqueCreatedClusters1.clear();

    for (int i = 0; i < PartitionAllInstances.size(); i++) {
        UniqueCreatedClusters1.add(
PartitionAllInstances.get(i).Cluster);
    }

}

    //store representatives from each part
  for (int l = 0; l < PartitionTotalMovedrepresentatives.size(); l++) {

TotalPartitions_Movedrepresentatives.add(PartitionTotalMovedrepresentatives.get(l));
  }

}


  for (int i = 0; i < TotalPartitions_Movedrepresentatives.size(); i++) {
```

```
                    numofCreatedClusters.add(i);
                    Totalrepresentatives.add(new
TotalCl(TotalPartitions_Movedrepresentatives.get(i).DataInst,
TotalPartitions_Movedrepresentatives.get(i).Cluster));
                    TotalMovedrepresentatives.add(new
TotalCl(TotalPartitions_Movedrepresentatives.get(i).DataInst,
TotalPartitions_Movedrepresentatives.get(i).Cluster));
                }

                Set<Integer> UniqueCreatedClusters = new
HashSet<Integer>(numofCreatedClusters);
                counter = 0;

                while( UniqueCreatedClusters.size() > m_NumClusters ) {
                        //begin always from the first
                        //find min distances for each point to all the other..
                        CalcMinDistance(TotalMovedrepresentatives.get(0),
TotalPartitions_Movedrepresentatives, Totalrepresentatives, TotalMovedrepresentatives, counter );
                        counter +=1;

                        UniqueCreatedClusters.clear();

                        for (int i = 0; i < TotalPartitions_Movedrepresentatives.size(); i++) {
                                UniqueCreatedClusters.add(
TotalPartitions_Movedrepresentatives.get(i).Cluster);
                        }
                }

                //begin categorize
                CategorizeFile(Totalrepresentatives);

        }

        public void CategorizeFile( List<TotalCl> tot_rep ) throws FileNotFoundException,
IOException{
                 //total file
                    Instances data;
                    try (
                            BufferedReader reader = new BufferedReader(
                                    new FileReader("TestData.arff"))) {data = new
Instances(reader);}
                        // setting class attribute
                        data.setClassIndex(data.numAttributes() - 1);
                        data.insertAttributeAt(new Attribute("Cluster"),
data.numAttributes());

                        double minDist = Integer.MAX_VALUE;
                        int rep = 0;  //holds instance with max distance
```

```
                    //predefine representatives
                    for (int j=0; j< data.numInstances(); j++){
                            //arxikopoihsh omadas cluster...
                            data.instance(j).setValue(data.numAttributes() - 1,
-1);
                            minDist = Integer.MAX_VALUE;
                            for (int i = 0; i < tot_rep.size(); i++) {
                                    //check if instance is already clustered
                                    double dist =
CategorizeMyDistanceFunction(data.instance(j), tot_rep.get(i).DataInst);
                                    if (dist < minDist) {
                                            minDist = dist;
                                            rep = i;
                                    }
                            }
                            data.instance(j).setValue(data.numAttributes() - 1,
tot_rep.get(rep).Cluster );
                    }

                //save into new file
                ArffSaver saver = new ArffSaver();
                saver.setInstances(data);

                saver.setFile(new File("./CureCategoryFile.arff"));
                //saver.setDestination(new  File("./CureCategoryFile.arff"));     //  **not**
necessary in 3.5.4 and later
                saver.writeBatch();
        }

        public double CategorizeMyDistanceFunction (Instance inst1, Instance inst2){
                double total_ = 0;
                for (int k = 0; k < inst1.numAttributes()-1; k++) {
                        double x =0;
                        x =  Math.pow(inst1.value(k) -  inst2.value(k), 2)  ;
                        total_ +=  x;
                }
                total_ = Math.sqrt(total_);
                return total_;
            }

        public double MyDistanceFunction (Instance inst1, Instance inst2){
                double total_ = 0;
                for (int k = 0; k < inst1.numAttributes(); k++) {
                        double x =0;
                        x =  Math.pow(inst1.value(k) -  inst2.value(k), 2)  ;
```

```
                    total_ += x;
                }
                total_ = Math.sqrt(total_);
                return total_;
        }

        public int CalcMinDistance(TotalCl instance, List<TotalCl> tot_instanses, List<TotalCl>
tot_rep, List<TotalCl> tot_moved, int counter ) {

                double minDist = Integer.MAX_VALUE;
                int rep = 0; //holds instance with min distance
                int repj = 0; //holds instance with min distance

                //predefine representatives, keeps indexes of total rep
                List<Integer> temp_rep = new ArrayList<Integer>();
                temp_rep.clear();

                for (int i = 0; i < tot_moved.size()-1 ; i++) {
                        //find distance between points
                        for (int j = i+1; j < tot_moved.size(); j++) {
                          int a = tot_moved.get(i).Cluster;
                          int b = tot_moved.get(j).Cluster;

                          if ( a!=b ) {
                                        double dist =
MyDistanceFunction(tot_moved.get(i).DataInst, tot_moved.get(j).DataInst);

                                        if (dist < minDist) {
                                                minDist = dist;
                                                rep = i;
                                                repj = j;
                                        }
                          }
                        }
                }

                int first_cl = tot_moved.get(rep).Cluster ; //holds cluster which changes, to
change alla instances of her to the new cluster

                List<TotalCl> representatives = new ArrayList<TotalCl>();
                representatives.clear();

              int changed_cluster = tot_moved.get(repj).Cluster;   //instance.Cluster;
              for (int i = 0; i < tot_instanses.size(); i++) {
                                if(     tot_instanses.get(i).Cluster      ==      changed_cluster     ||
tot_instanses.get(i).Cluster == first_cl  ){
```

```
                                tot_instanses.get(i).Cluster =  changed_cluster ;
                                representatives.add(tot_instanses.get(i));
                        }
                }

                //remove from rep the new cluster and the moved rep
                //keep indexes of rep/moved representatives to be remove
                List<Integer> remove_index ;
                remove_index = new ArrayList<Integer>();
                remove_index.clear();

                for (int i = 0; i < tot_rep.size(); i++) {
                        if( tot_rep.get(i).Cluster == changed_cluster || tot_rep.get(i).Cluster
== first_cl  ){
                                remove_index.add(i);
                        }
                }

                for (int i = 0; i < remove_index.size(); i++) {
                        int rv = remove_index.get(i)-i;
                        tot_moved.remove(rv);
                        tot_rep.remove(rv);
                }

                findRepresentatives(representatives, tot_rep, tot_moved , counter ) ;

                return rep;
        }

        private void findRepresentatives( List<TotalCl> new_cl, List<TotalCl> tot_rep ,
List<TotalCl> tot_moved , int counter) {

                int representativesPercent = 5;
                int representativesCount = (int) ( new_cl.size()
*representativesPercent/100);
                representativesCount = (representativesCount > 1) ?
representativesCount : 1;

                findCentroid1(new_cl);

                //finding the center of cluster -->>find representatives by counting the
farthest distance of instances from the center
                //create arraylist to keep representatives, for massive loop -->> use rep to
find clusters

                //find max distance
                int rep1 = 0;
```

```
                double maxDist = 0;
                double dist_cen_rep = 0;
                List<TotalCl> new_representatives = new ArrayList<TotalCl>();
                List<TotalCl> MovedRepresentatives = new ArrayList<TotalCl>();
                new_representatives.clear();
                MovedRepresentatives.clear();

                while (new_representatives.size() < representativesCount)
                {
                        maxDist = 0;
                        dist_cen_rep=0;

                        for (int i = 0; i < new_cl.size(); i++) {
                                double dist = MyDistanceFunction(findCentroid,
new_cl.get(i).DataInst);
                                if (dist > maxDist ) {
                                        maxDist = dist;
                                        dist_cen_rep = dist;
                                        rep1 = i;
                                }
                        }
                        //cendroid of cluster to moved for alpha
                        //moved rep
                        TotalCl Mv = new TotalCl(findMovedRep (findCentroid ,
new_cl.get(rep1), dist_cen_rep, counter), new_cl.get(rep1).Cluster ) ;
                        MovedRepresentatives.add(Mv);

                        new_representatives.add(new_cl.get(rep1)); //max distance -->>
representatives
                        new_cl.remove(rep1); //delete it for not get the same point
                }

                for (int i=0; i< MovedRepresentatives.size(); i++) {
                        tot_rep.add(new_representatives.get(i));
                        tot_moved.add(MovedRepresentatives.get(i));
                }
        }

        private Instance findMovedRep( Instance Centroid, TotalCl Rep, double dist, int counter ) {
                //move for alpha representative to centroid
                Instance MovedRep = (Instance)(Rep.DataInst) ;

                //find direction of movenment
                for (int k = 0; k < Rep.DataInst.numAttributes(); k++) {
                        Attribute att = Rep.DataInst.attribute(k);
                        //clear content of instance, each attribute..
```

```
                    MovedRep.setValue(att, 0);

                    double total_= 0;
                    double rep_  = 0;
                    double cent_  = 0;

                    rep_ +=  Rep.DataInst.value(k);
                    cent_ +=  Centroid.value(k);
                    total_ = rep_*(1-alpha)+cent_*alpha;
                    MovedRep.setValue(att, total_);
                }
                return MovedRep;
        }

        private Instance findCentroid1( List<TotalCl> cl_instanses ) {
                //calculate the center of all points of cluster to find reprecentatives -->>
most far points
                Instance Centroid = new Instance(cl_instanses.get(0).DataInst) ;

                for (int k = 0; k < cl_instanses.get(0).DataInst.numAttributes(); k++) {
                    Attribute att = cl_instanses.get(0).DataInst.attribute(k);
                    //clear content of instance, each attribute..
                    Centroid.setValue(att, 0);
                    double mesh_timh = 0;
                    for (int nm = 0; nm < cl_instanses.size(); nm++) {
                            Instance a = cl_instanses.get(nm).DataInst;
                            mesh_timh +=  a.value(k);
                      }
                      if (cl_instanses.size()>0) {
                             mesh_timh = mesh_timh / cl_instanses.size();
                      }
                      Centroid.setValue(att, mesh_timh);
                }
                findCentroid = Centroid;
                return Centroid;
        }

        /**
         * @param args the command line arguments
         * @throws java.lang.Exception
         */
        public static void main(String[] args) throws Exception {
                JavaApplication7 cure=new JavaApplication7();
                cure.StartCure();
        }
```

```
	public void StartCure()throws Exception {
		{
			Instances data;
			try (BufferedReader reader = new BufferedReader(new
FileReader("TestDataSample.arff"))) {
				data = new Instances(reader);
			}
			// setting class attribute
			data.setClassIndex(data.numAttributes() - 1);
			buildClusterer(data);

		}
	 }

	private  class  TotalCl {
		private Instance DataInst;
		private Integer Cluster;

		public TotalCl(Instance DataInst, Integer Cluster) {
			this.DataInst = DataInst;
			this.Cluster = Cluster;
		}
	}
private class clustering_results {
		String cluster_description;
		int cluster_ins;
		float cluster_mean;

		public clustering_results(String cluster_description, int cluster_ins, float
cluster_mean) {
		  this.cluster_description = cluster_description;
		  this.cluster_ins = cluster_ins;
		  this.cluster_mean = cluster_mean;
		}
	}
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Agrawal, R., and Psaila, G. 1995. Active Data Mining. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), 3–8. Menlo Park, Calif.: American Association for Artificial Intelligence.

[2] Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, I. 1996. Fast Discovery of Association Rules. In Advances in Knowledge Discovery and Data Mining, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 307–328. Menlo Park, Calif.: AAAI Press.

[3] Piatetsky-Shapiro, G. 1991. Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop. AI Magazine 11(5): 68–70.

[4] W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus, Knowledge Discovery in Databases: An Overview, in G. Piatetsky-Shapiro and W. J. Frawley (eds.), Knowledge Discovery in Databases, AAAI/MIT Press, 1991, 1−27.

[5] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.-H., Steinbach, M., Hand, D. J., and Steinberg, D. 2007. “Top 10 Algorithms in Data Mining,” Knowledge and Information Systems (14:1), pp. 1-37.

[6] Arthur Samuel: Pioneer in Machine Learning, Σεπ. 2015 http://infolab.stanford.edu/pub/voy/museum/samuel.html

[7] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “The KDD process of extracting useful knowledge”, Volumes of data Communications of the ACM, 39(11):27–34, November 1996.

[8] Dakshi Agrawal and Charu C. Aggarwal. 2001. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (PODS '01). ACM, New York, NY, USA, 247-255.

[9] Paul S. Bradley, Usama Fayyad, and Cory Reina. Scaling Clustering Algorithms to Large Databases. In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, (KDD), pages 9–15, New York, NY, USA, 27–31 August 1998. AAAI Press.

[10] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. CACTUS: Clustering Categorical Data Using Summaries. In Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, (KDD), pages 73–83, San Diego, CA, USA, 15–18 August 1999. ACM Press.

[11] Fabrizio Sebastiani, “Machine learning in automated text categorization” ,ACM Computing Surveys(CSUR),Vol. 34, Issue 1,March 2002

[12] S.B.Kotsiantis Department of Computer Science and Technology University of Peloponnese, Greece “Supervised Machine Learning: A Review of Classification Techniques Informatics 31(2007) 249-268

[13] Dharminder Kumar, Suman” Analysis of Various Data A Review, International Journal of Computer Applications (0975 – 8887) October 2011

[14] Jiawei Han and Michelle Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2001.

[15] Margaret H.Dunham, επιμέλεια: Βασίλης Βερύκιος, Γιάννης Θεοδωρίδης, "Data Mining Εισαγωγικά και προηγμένα θέματα εξόρυξης γνώσης από δεδομένα".

[16] A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data clustering: a review. *ACM Comput. Surv.* 31, 3 (September 1999), 264-323.

[17] Paul S. Bradley, Usama Fayyad, and Cory Reina. Scaling EM (Expectation-Maximization) Clustering to Large Databases. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA, USA, October 1999.

[18] Martin Ester, Hans-Peter Kriegel, Joer ¨ g Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In

Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, (KDD), pages 226–231, Portland, OR, USA, 2–4 August 1996. AAAI Press.
[19] Brian S. Everitt. Cluster Analysis. Edward Arnold, 1993.
[20] Anil K. Jain and Richard C. Dubes. Algorithms for Clustering Data. Prentice-Hall, 1988.
[21] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data* (SIGMOD '99). ACM, New York, NY, USA, 49-60.
[22] Fisher, D. (1987b). Knowledge acquisition via incremental conceptual clustering. Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.
[23] W. Wang, J. Yang and R.Muntz, "STING: A statistical information grid approach to spatial data mining" Proceedings of the International Very Large Databases Conference, pages 186-195, 1997.
[24] S Ryszard Michalski, G Jaime Carbonell, and M Tom Mitchell *(Eds.). 1986.* "*Machine Learning an Artificial Intelligence Approach Volume II"* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
[25] Machine Learning with WEKA, Σεπ.2015
http://csed.sggs.ac.in/csed/sites/default/files/WEKA%20Explorer%20Tutorial.pdf
[26] Citing Weka
http://www.cs.waikato.ac.nz/ml/weka/citing.html
[27] Thair Nu Phyu,proceedings of the Internationalulticonference of Engineers and Computer Scientists 2009 Vol I,IMECS 2009,March18-20,2009 Hong Kong"Survey of Classification Techniques in Data Mining"
[28] Rohit Arora, Suman" Comparative Analysis of Classification Algorithms on Different Datasets using WEKA International Journal of Computer Applications (0975 – No.13, September 2012
[29] B. Chaudhari, M. Parikh, A comparative study of clustering algorithms using weka tools. Int. J. Appl. Innovation Eng Manage (IJAIEM). **1**(2) (2012)
[30] B. Chaudhari, M. Parikh, A comparative study of clustering algorithms using weka tools. Int. J. Appl. Innovation Eng Manage (IJAIEM). **1**(2) (2012)
[31] Sharma,N., Bajpai,A., Litoriya,R., "Comparing the various clustering algorithms of Weka tool",International Journal of Emerging Technology and Advanced Engineering ,2(5),2012.
[32] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2(2), 1987.
[33] Sander J., Ester M., Kriegel H.-P., Xu X.:"Dens@-Based Clustering in Spatial Darabases: The Algorirhm GDBSCAN and its Applications ", will appear in: Data Mining and Knowledge Discovery, Kluwer Acedemic Publishers, Vol. 2, 1998.
[34] Sharma,N., Bajpai,A., Litoriya,R., "Comparing the various clustering algorithms of Weka tool",International Journal of Emerging Technology and Advanced Engineering ,2(5),2012.
[35] D. Pelleg and A. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters", Proc. 17th Int. Conf. Machine Learning (ICML',00), pp.727 -734 2000.
[36] M. Ankerst, M. Breunig, H. Kriegel and J. Sander, "Optics: Ordering Points to Identify the Clustering Structure," Proc. SIGMOD, 1999.
[37] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data* (SIGMOD '98), Ashutosh Tiwary and Michael Franklin (Eds.). ACM, New York, NY, USA, 73-84
[38] Pelekis, N.; Ntrigkogias, C.; Tampakis, P.; Sideridis, S.; and Theodoridis, Y. 2013. Hermoupolis: A trajectory generator for simulating generalized mobility patterns. In Machine Learning and Knowledge Discovery in Databases. 659–662.