Διάβασα το αρχείο μου, το κανονικοποίησα και το τροποποίησα έτσι ώστε να υπολογίσω τα κέντρα.

1) Όρισα ένα κατώφλι 0.2, δηλαδή αν το καινούργιο SSE είναι πάνω από το 80% του παλιού κράτα τα προηγούμενα ακριβώς κέντρα. Ξεκίνησα με 2 κέντρα υπολόγισα το SSE και το αποθήκευσα. Μετά έτρεξα τον K_Means για 3 κέντρα, υπολόγισα το SSE και το σύγκρινα με το SSE για τα 2 κέντρα. Αν το SSE ήταν κάτω από το 80% του παλιού συνέχιζα και έβρισκα το SSE για 4 κέντρα και το σύγκρινα με το κόστος για 3 κέντρα. Και αυτό συνέχιζε μέχρι να ικανοποιούταν η συνθήκη. Για να το εξηγήσω καλύτερα πχ αν το παλιό μου SSE ήταν 100 και το καινούργιο ήταν 60 συνέχιζα τον αλγόριθμο. Αν όμως το παλιό μου SSE ήταν 100 και το καινούργιο 85 ο αλγόριθμος σταματούσε και κρατούσε τα κέντρα με SSE=100.
2) Εκμεταλλευόμενος πως οι αποστάσεις των κέντρων από τα σημεία ακολουθούν κανονική κατανομή, με μια συνάρτηση (centers) υπολόγισα το κέντρο που ανήκει κάθε σημείο και το κοντινότερο κέντρο στο σημείο αυτό και το επέστρεφα σε ένα RRD το οποίο σαν πρώτο στοιχείο είχε ένα vector με τα x και y του σημείου και σαν δεύτερο στοιχείο είχε το αποτέλεσμα της συνάρτησης centers. Αυτό ήταν το κύριο μου RDD (**main_RDD**), εδώ είχα το μεγαλύτερο μέρος της πληροφορίας που χρειαζόμουν για να βρω τα AI και BI του κάθε σημείου και έπειτα της κάθε συστάδας μου. Με βάση το **main_RDD** μου έφτιαξα 2 νέα RDD το **primary_key_A** (το οποίο είχε σαν κύριο κλειδί σε ποια συστάδα ανήκει το σημείου μου, και σαν value το vector του σημείου και την κοντινότερη του συστάδα) και το **primary_key_B** (το οποίο είχε σαν κύριο κλειδί το ποια είναι η κοντινότερη συστάδα του σημείου αυτού, και σαν value το vector του σημείου και σε ποια συστάδα ανήκει). Μετα έκανα join το **primary_key_A** με τον εαυτό του έτσι ώστε να μου φτιάξει ζευγάρια με τα σημεία τα οποία ανήκουν στην ιδιά συστάδα για να μπορέσω να υπολογίσω αποστάσεις για το **AI**. Αμέσως μετα έκανα join το **primary_key_B** με το **primary_Key_A** έτσι ώστε να μου βρει τα ζευγάρια που χρειάζεται να έχω για να υπολογίσω το **BI**. Στα

επόμενα βήματα (τα οποία είναι κοινά και για το **AI** και για το **BI**) βάζω σαν κύριο κλειδί την συστάδα στην οποία ανήκει το κάθε σημείο και το vector του σημειου (στο **BI** βάζω σαν κύριο κλειδί, το vector και την συστάδα, που προήλθε από το **primary_key_B**) και υπολογίζω τις αποστάσεις σε κάθε ζευγάρι που προέκυψε. Μετα με την reduceByKey βρισκω το AI και το BI του κάθε σημείου ξεχωριστά (στο **BI** το reduceByKey υπολογίζει και το πόσες φορές εμφανίστηκε ένα στοιχειο) . Στο τελευταίο βήμα κάνω join το **AI** και το **BI** και υπολογίζω τον συντελεστή σιλουέτας του κάθε σημείου και θέτω σαν κύριο κλειδί την συστάδα που ανήκει το κάθε σημείο. Με την reduceByKey βρισκω το άθροισμα για την κάθε συστάδα ενώ παράλληλα υπολογίζω και το μέγεθος της συστάδας. Τέλος διαιρω το άθροισμα που βρήκα με το μέγεθος της κάθε συστάδας και αυτός είναι ο συντελεστής σιλουέτας της συστάδας.

3) Στον εντοπισμό των ανωμαλιών εκμεταλλεύτηκα ότι το κέντρο κάθε cluster είναι και η μέση τιμή του και πως οι αποστάσεις των κέντρων από τα σημεία ακολουθούν κανονική κατανομή, και βρήκα την τυπική απόκλιση (σ) μόνο για ένα cluster (και για τον άξονα x και για τον άξονα y) διότι όλες οι συστάδες έχουν κοινές παραμέτρους. Μετα όρισα ένα κατώφλι=3.5 και είπα πως αν η απόλυτη τιμή της απόστασης του σημείου από το κέντρο ξεπερνάει το 3.5σ (είτε για τον άξονα x, είτε για τον άξονα y, είτε και για τους 2), τότε το σημείο αυτό είναι ακρότατο. Το κατώφλι το όρισα μετα από αρκετές δοκιμές και βγάζει 30 ακρότατα (outliers).

*Όσα ονόματα RDDs χρησιμοποιώ και στο πρόγραμμα και είναι σημαντικά τα τόνισα με **Bold.**

**Το διάβασμα και τον καθαρισμό του αρχείου τον κάνω με sql.dataframe, καθώς λειτουργεί πιο γρήγορα από ότι αν το κάνω με RDD.

***Ο χρόνος εκτέλεσης του προγράμματος είναι περίπου 4.5-5.5 λεπτά σε σταθερό υπολογιστή με AMD FX 8320 και 8GB ram με ταχύτητα στα 1333 MHz.

****Τα σημεία κάθε κέντρου, το μέγεθος του κάθε cluster όπως και την τυπική απόκλιση τα αποθήκευσα σε πίνακες έτσι ώστε να μπορώ να τα καλέσω όταν τα χρειαζόμουν μέσα σε κάποια διεργασία της spark (map,reduce,..) διότι τα RDDs δεν με άφηνε να τα καλέσω.

*****Χρησιμοποίησα Scala 2.10.7, Spark 2.2.0, Spark-sql 2.2.0 και Spark-mllib 2.2.0.

******Παρατήρησα ότι οι τυπικές αποκλίσεις των cluster έχουν μια ανεπαίσθητη διαφορά, παρόλα αυτά δεν επηρεάζει το πως βγαίνουν τα outliers όταν έχω τα σωστά κέντρα αλλά βγάζει λάθος outliers όταν δεν έχω τα σωστά κέντρα, για αυτό και υπολογίζω την τυπική απόκλιση για κάθε cluster ξεχωριστά. Έχω όμως τον κώδικα που υπολογίζει μόνο μια τυπική απόκλιση σε σχόλια.