ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΕ041 - ΠΛΕ081: Διαχείριση Σύνθετων Δεδομένων

(ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2022-23)

ΕΡΓΑΣΙΑ 1 - Ιστογράμματα

Βλαχόπουλος Λάμπρος, ΑΜ : 2948

Περιεχόμενα

1)

LoaderCsv

Η κλάση LoaderCsv είναι υπεύθυνη για την φόρτωση των δεδομένων από αρχείο csv.

Αυτό που κάνει είναι να διαβάζει γραμμή γραμμή το αρχείο και να δημιουργεί για κάθε γραμμή μια λίστα με τα tokens κάθε γραμμής split-αρισμενα με βάση το delimiter. Αφού υπολογιστεί η θέση του πεδίου που θέλουμε να πάρουμε πχ (Income) τα δεδομένα παίρνουμε από τη λίστα με βάση τη θέση αυτή την τιμή που υπάρχει κάθε φορά και δημιουργούμε αντικείμενο τύπου Income. Στη συνέχεια προσθέτουμε αυτό το Income αντικείμενο σε μια λίστα ενός Manager που χρησιμοποιούμε βοηθητικά για να ομαδοποιήσουμε τα Incomes που εξάγουμε από το αρχειο. Εδώ να πουμε ότι όταν δεν κρατείται τιμή για το πεδίο που θέλουμε συμπληρώνουμε την λίστα με Not Info.

Πχ Έστω ότι έχουμε την γραμμή

CensusTract,State,County,TotalPop,Men,Women,Hispanic,White,Black,Native,Asian,Pacific,Citizen,Income , …

1001020100,Alabama,Autauga,,940,,0.9,,7.7,0.3,0.6,0.0,1503,61838.0,…

Η λίστα θα είναι της μορφής

[1001020100,Alabama,Autauga,Not info,940,Not info,0.9,Not info,7.7,0.3,0.6,0.0,1503,61838.0,…

,…]

Και αφού βρούμε την θέση του πεδίου που θέλουμε να κρατήσουμε δεδομένα πχ του income =14

Παίρνουμε από τη λίστα value=list.get(14-1);

Income=New Income(value)

List.add(Income) //manager

Income

Η Κλάση Income αντιπροσωπεύει τα Values που κρατάμε κάθε φορά.

Κατά την δημιουργία ενός αντικειμένου ελέγχουμε αν υπάρχει τιμή ή όχι στο αρχείο για αυτό.

Αν υπάρχει η τιμή ισούται με ότι υπάρχει στο αρχείο, αλλιώς βάζουμε αυθαίρετα ένα errorCode =-100.0 σαν value που θα μας βοηθήσει αργότερα να τα αγνοήσουμε.

Income Manager

- Η κλάση IncomeManager διαχειρίζεται τα αντικείμενα Income και τα εισάγει σε μια λίστα μόνο αν η τιμή value είναι διαφορετική από το errorCode που όρισα. Η διαδικασία εισαγωγής στη λίστα γίνεται κατά το διάβασμα γραμμής - γραμμής του αρχείου και δεν επαναλαμβάνω να διατρέξω όλα τα φορτωθεν δεδομένα για να τα διαχωρίσω.
- Επίσης με την κλάση αυτή sort -αρουμε τα δεδομένα στη λίστα με χρήση compactor με βάση το πεδίο IncomeValue.
- Επιπλέον βρίσκω το αντικειμενο με την max kai min τιμή από τη λίστα καθώς έχουμε σορταρισμενη λίστα με τις τιμές και δεν κάνουμε χρήση αλγορίθμων εύρεσης max και min.

EquiWidth

calculatePairs(): Βρίσκει τα Bins που προκύπτουν από τα δεδομένα. Προσθέτω το τελευταίο Bin με όσα δεδομένα έχουν απομείνει δηλαδή το τελευταίο bin μπορεί ανάλογα τα δεδομένα να έχει διαφορετικό binSize από τα υπόλοιπα και το υλοποίησα έτσι με σκοπό να μην έχω απώλεια δεδομένων. Τα ζευγάρια τιμών που προκύπτουν εισάγονται σε μια λίστα η οποία θα είναι ταξινομημένη.

findValuesInPair(): Βρίσκει τις τιμές που υπάρχουν σε κάθε bin. Εδώ αυτό που κάνω είναι να βρω σε πιο εύρος τιμών βρίσκεται η κάθε τιμή που διάβασα από το αρχείο χωρίς να ελέγξω αν αυτή βρίσκεται στα εύρη τιμών. Δηλαδή δεν ελέγχω πχ αν πχ η τιμή 2500 είναι στο (1000,2000) με ελέγχους τύπου if (2500<2000 && 2500>1000) αλλά αυτό που κάνω είναι να βρω κατευθείαν σε ποιο εύρος διαστήματος βρίσκεται η τιμή χωρίς να ξέρω ποιο είναι αυτό. Βρισκω την θέση του ζευγαριού στη λίστα που υπάρχει αυτή η τιμή. Ορίζω έναν πίνακα μεγέθους με τον αριθμό των bins , έτσι αντιστοιχίζω το κάθε κελί του πίνακα με το κάθε εύρος τιμών που έχω στην λίστα, και αυξάνω την τιμή του κελιού κατά 1 κάθε φορά.

Estimate(a,b) : Η βασική της λειτουργία είναι να βρει αρχικά σε πιο Bin ανήκουν α,β. Ανάλογα σε πιο Bin ανήκουν θα υπολογιστούν οι εγγραφές για το α μετά για το β και στο τέλος θα κληθεί μια εσωτερική μέθοδος που θα υπολογίσει τις ενδιάμεσες εγγραφές και το άθροισμα όλων θα δώσει το αποτέλεσμα.

actulResults (α,β): Υπολογίζει τα πραγματικά δεδομένα από τις εγγραφές που διαβάστηκαν.

EquiDepth

calculatePairs(): Βρίσκει τα Bins που προκύπτουν από τα δεδομένα. Προσθέτω το τελευταίο Bin με όσα δεδομένα έχουν απομείνει δηλαδή το τελευταίο bin μπορεί ανάλογα τα δεδομένα να έχει διαφορετικά numtuples από τα υπόλοιπα και το υλοποίησα έτσι με σκοπό να μην έχω απώλεια δεδομένων. Τα ζευγάρια τιμών που προκύπτουν εισάγονται σε μια λίστα η οποία θα είναι ταξινομημένη. Επίσης ξέρω σίγουρα ότι το τελευταίο bin όταν έχει περισσότερες τιμές από τα υπόλοιπα αυτές οι extra τιμές θα είναι σίγουρα λιγότερες από τον αριθμό των bins αλλιώς σε διαφορετική περίπτωση οι τιμές οι extra θα μπορούσαν να μοιραστούν όλες ακριβώς στα bins .

findValuesInPair(): Εδώ τα πράγματα είναι απλά καθώς κάθε bin θα έχει τον ίδιο αριθμό από numtuples. Εκτός από το τελευταίο που θα έχει ότι περισσεύει από το αν αφαιρέσουμε από όλες τις εγγραφές που διαβάσαμε τα (bins-1)*numtuplesPerBin.

Estimate(a,b): Η βασική της λειτουργία είναι να βρει αρχικά σε πιο Bin ανήκουν α,β. Ανάλογα σε πιο Bin ανήκουν θα υπολογιστούν οι εγγραφές για το α μετά για το β και στο τέλος θα κληθεί μια εσωτερική μέθοδος που θα υπολογίσει τις ενδιάμεσες εγγραφές και το άθροισμα όλων θα δώσει το αποτέλεσμα. Εδώ να πω ότι επειδή τα bins δεν έχουν ίδιο binSize δε μπορώ να βρω κατευθείαν σε πιο bin είναι μια τιμή όπως στην περίπτωση **EquiWidth.** Υπολογίζω σε ποιο bin βρίσκονται τα α,β με binnarySearch() στην λίστα που κρατάω τα ζευγάρια τιμών καθώς αυτή είναι ταξινομημένη.

actulResults(α,β): Υπολογίζει τα πραγματικά δεδομένα από τις εγγραφές που διαβάστηκαν.

Main : Δημιουργεί τα απαραίτητα αντικείμενα και καλεί τις απαραίτητες μεθόδους . Εκεί είναι και υλοποιημένο το τεστ για το $2^{ο}$ μερος.

Παρατήρηση : Ο κώδικας θα μπορούσε να υλοποιηθεί με ακόμα περισσότερη αντικειμενοστραφή τρόπο αλλά δεν θεώρησα απαραίτητο ως προς το σκοπό του μαθήματος. Αυτό που είχα σκοπό να κάνω είναι μια abstract κλάση που θα κληρονομούσαν τα είδη διαγραμμάτων κάποια πράγματα ώστε να μην είναι μακροσκελή και έχουν κοινό κώδικα στις κλάσεις που είναι υπεύθυνες για τα διαγράμματα.

2)

Για την μελέτη αυτό που έκανα είναι να δημιουργήσω κάποια αρχικά κάποια ranges. Δημιούργησα HashMap's με δεδομένα της μορφής

1) **{500,1000} {1500 ,2000}…**
2) **{1000,2000} {2000 ,3000}…**
3) **{2500,5000} {5000 ,7500}…**
4) **{5000,10000} {10000 ,15000}…**
5) **{10000,20000} {20000 ,30000}…**
6) **{20000,40000} {40000 ,60000}…**
7) **{30000,60000} {60000 ,90000}…**
8) **{50000,100000} {100000 ,150000}…**
9) **{80000,160000} {160000 ,240000}…**
10) **{100000,200000}**
11) **{150000,300000}**

Στην συνέχεια παράγω και 500 τυχαίες τιμές για κάθε ζεύγος που έχω συνολικά και καλώ τις estimate(a,b) με όλες τις τιμές που προκύπτουν. Στο κώδικα υπολογίζω το σφάλμα που υπάρχει από τα actual που υπάρχουν στα αρχεία και ενημερώνω τις μεταβλητές κάθε φορά ανάλογα.

Εν τέλη για τα δεδομένα αυτά προκύπτει ότι καλύτερο είναι το πρώτο διάγραμμα με τα συγκεκριμένα τεστ. (EquiWidth).

Να σημειώσω τα τεστ αφορούν μόνο το αρχείο που μας δόθηκε δεν είναι γενικά τεστ που τσεκάρουν αρχεία φόρτωσης.