Πολυπρακτορικα Συστηματα ΠΛΗ 512

Αναφορά Προγραμματιστικής Εργασίας

Μανεσιώτης Αθανάσιος Α.Μ. : 2017030085
Μονογυιός Αντώνιος Α.Μ. : 2017030013
Κλιούμης Γεώργιος Α.Μ. : 2017030116

Οδηγίες Εκτέλεσης

Για την κατασκευή της προγραμματιστικής εργασίας έγινε χρήση της python 3.8. Τα packages τα οποία χρησιμοποιήθηκαν είναι τα εξής: random, threading, pathlib, xlrd, xlsxwriter, pandas, numpy, multiprocessing. Ο κώδικας του project συμπηκνώθηκε σε ένα αρχείο main.py, μαζί με τις συναρτήσεις που χρειάζονται για τον χειρισμό των αρχείων, εκτύπωσης στατιστικών και εκτέλεσης αλγορίθμων (βρίσκονται στο αρχείο με αυτή την σειρά από πάνω προς τα κάτω). Σχολιασμός της λειτουργικότητας της εκάστοτε συνάρτησης βρίσκεται πάνω από την δήλωσή της, ενώ περαιτέρω σχολιασμός της λογικής βρίσκεται κατά περίπτωση σε inline σχόλιο. Για το control sequence του προγράμματος κάτω από τα Imports ορίζονται οι εξής μεταβλητές:

- SPEND_TIME_WAITING: boolean Array όπου το κάθε index αντιστοιχεί και σε ένα Task της εργασίας ανά σειρά (το Task 2 έχει a και b). Αν λοιπόν χρειάζεται να παρθούν τα αποτελέσματα του Task 2a, ορίζεται SPEND_TIME_WAITING = [False, True, False, False, False, False]. Σημειώνεται ότι το Task 2a πρέπει να τρέξει μόνο του διότι το copeland method χρησιμοποεί threads τα οποία δεν έχουν γίνει join στο main thread, οπότε θα συνεχίσει η εκτέλεση του προγράμματος στον rav και έτσι τα αποτελέσματα της κονσόλας θα μπερδευτούν
- EPSILON: ορίζει το μικρότερο δυνατό rating που μπορεί να έχει ένας χρήστης για κάποιο item
- IMPORT_MATRICES: boolean μεταβλητή που καθορίζει εάν οι πίνακες prefList, r, καθώς και οι πίνακες που αναφέρουν τα budget των χρηστών και τις τιμές των items, θα φορτωθούν από τον φάκελο του project (True) ή θα αποθηκευτούν σε αυτόν (False). Στην πρώτη εκτέλεση του προγράμματος είναι by default False ώστε να δημιουργηθούν και να αποθηκευτούν οι προαναφερθείς πίνακες, ενώ συστήνεται να γίνει True για τις μετέπειτα εκτελέσεις, μιας και είναι μία δύσκολη υπολογιστικά εργασία.
- KENDALL_TAU_GROUP_SIZE: μεταβλητή που καθορίζει το μέγεθος των 100 group στα οποία θα υπολογιστεί η Kendall_τ για την δημιουργία divergent και similar groups.

Τα αποτελέσματα του εκάστοτε Task φαίνονται στην κονσόλα της python. Τα Datasets του project καθώς και οι cached πίνακες (στην περίπτωση που δημιουργηθούν) βρίσκονται μέσα στον φάκελο Datasets.

Task 1

Πρώτο βήμα της προγραμματιστικής εργασίας ήταν η δημιουργία μία λίστας προτιμήσεων(prefernce list) των χρήστών (users) για τα αντικείμενα(items). Για να μπορέσει να πραγματοποιηθεί η σύγκριση μεταξύ των Gaussian μεταβλητών, που περιγράφουν τους χρήστες και τα items, αξιοποιήθηκε το μέτρο ομοιότητας

Kullback-Leibler, το οποίο περιγράφεται απ΄την εξής εξίσωση:

$$KL(u||i)$$
$$= \frac{1}{2}\log\left|\sum_u^{-1}\sum_i\right| + \frac{1}{2}tr\left(\left(\sum_u^{-1}\sum_i\right)^{-1}\right) - \frac{D}{2} + \frac{1}{2}(\mu_u - \mu_i)^T\sum_i^{-1}(\mu_u - \mu_i)$$

όπου $\sum_u, \mu_u, \sum_i, \mu_i$ είναι παράμετροι κατανομής και tr() είναι το ίχνος του αντίστοιχου πίνακα.

Όταν η KL παίρνει μικρές τιμές, αυτό σημαίνει ότι υπάρχει αρκετή ομοιότητα μεταξύ του χρήστη και του item. Ενώ αντίθετα, όσο πιο μεγάλες τιμές πάρει, τόσο πιο ανόμοια είναι μεταξύ τους. Αξίζει να σημειωθεί ότι σε ορισμένες περιπτώσεις η KL έπαιρνε αρνητικές τιμές, με αποτέλεσμα να χρειαστεί να γίνει προσαρμογή της τιμής στο 0.

Τέλος, με βάση την KL που ορίσθηκε παραπάνω, υπολογίζεται το τελικό rating που αντιπροσωπεύει την σχέση του χρήστη με το εκάστοτε αντικείμενο. Τα αποτελέσματα αποθηκεύονται σε ένα excel(preferenceList), προκειμένου να μπορούν να είναι προσβάσιμα και στην συνέχεια της άσκησης.

Η εξίσωση που παράγει τα ratings είναι η εξής:

$$r = r_{max} - \frac{KL(u||i)}{r_{max}}$$

όπου r_{max} η μέγιστη τιμή που μπορεί να δώσει ένας χρήστης σε ένα αντικείμενο. $r_{max} = 10$

Τέλος παρουσιάζεται λίστα με το top-10 για τους πρώτους 50 χρήστες.

```
Top 10 items for every user(descending)
User:  0 [ 43. 431. 335.  98.  54. 479. 463. 150. 238. 426.]
User:  1 [286. 130. 449. 121. 288. 201.  42. 181. 164. 321.]
User:  2 [ 14. 252. 453. 331. 423. 338. 414. 173. 437. 428.]
User:  3 [428.  30. 383.  66. 179.  68. 279.   0.  69. 235.]
User:  4 [162.  36. 447.  92. 165. 355. 280. 402. 350.  89.]
User:  5 [ 67. 495.  75. 451.  39. 426. 473. 165. 209.   8.]
User:  6 [306. 228. 160. 294. 140. 119. 159. 371. 267.   0.]
User:  7 [146. 255. 369.  54. 396.  93. 360. 301. 281. 239.]
User:  8 [213. 283. 161. 395. 481. 208. 317. 183.  93. 337.]
User:  9 [ 52. 211. 201. 455. 164. 485. 198. 166. 218. 404.]
User: 10 [217. 148. 435. 304. 345.  66. 169.  94. 254. 429.]
User: 11 [229.   7. 164.  72.  60.  42. 274. 220. 289. 181.]
User: 12 [203. 230. 397. 185. 393. 212. 157. 159.   6. 462.]
User: 13 [232. 203. 242.  78. 433. 285. 230. 263. 393. 310.]
User: 14 [ 39. 483. 220.  82. 472.  61. 325. 401.  41. 257.]
User: 15 [ 57. 163.  45. 101. 206.  52.  49. 172. 248. 269.]
User: 16 [264. 369. 145. 101. 281. 146.  98. 453. 414. 191.]
User: 17 [324. 437. 331. 446. 154. 350. 489. 200. 368. 478.]
User: 18 [200. 154. 489. 486. 416. 281. 432. 146. 186. 305.]
User: 19 [353. 349. 254.  87. 138. 428. 394. 235.  68. 220.]
User: 20 [264. 151. 206. 144. 495. 438.  45. 209.  49. 101.]
User: 21 [432. 368.  34. 123. 439. 446.  62. 305. 435. 377.]
User: 22 [321. 274. 130. 427. 286. 318.  72. 449. 494. 409.]
User: 23 [451. 463. 209. 493.  98. 453. 245. 495. 143. 371.]
```

(a)

```
User: 24 [171.  80. 311.  59. 336. 302.  11. 320. 408. 445.]
User: 25 [302.  71. 233.  17. 382. 449. 346. 337. 374.  11.]
User: 26 [ 62.  64. 292.  84. 446. 377. 391. 318. 122. 300.]
User: 27 [232. 242. 378.  57. 259. 310. 108.  78. 174. 314.]
User: 28 [ 16. 406.  85. 290.   1. 444. 441. 270. 492. 292.]
User: 29 [377. 240. 149.  34. 292. 350. 113. 402. 464. 478.]
User: 30 [276.  25. 177. 220. 262. 289.  77.  28. 130.   3.]
User: 31 [402. 149.   8. 240. 490. 350. 366. 113. 377. 119.]
User: 32 [402.  92. 270. 280. 162. 448.  22. 219. 490. 408.]
User: 33 [111. 192. 197. 190. 183.  99. 178. 283. 317.  76.]
User: 34 [250. 118. 275. 293. 317. 341. 354. 483. 360. 311.]
User: 35 [299. 106. 139. 357. 314. 452. 462. 204. 180.  69.]
User: 36 [113. 478. 292.   4. 365.  62. 295. 240. 361. 475.]
User: 37 [405. 418. 159. 359. 185. 452. 131. 299. 120.   6.]
User: 38 [394. 102. 320. 492. 198. 336.  68. 408.   8.  83.]
User: 39 [  4. 295. 375. 392. 261. 144. 191. 308. 272. 145.]
User: 40 [470.  48. 327.   9. 234. 439. 298.  25. 380.  20.]
User: 41 [201. 211. 332. 449. 320. 226. 198. 302. 374. 287.]
User: 42 [110. 399. 373. 209. 143.  75.  96.  38. 227.  37.]
User: 43 [ 17. 424. 444. 449. 121.  12. 226. 135.  71. 365.]
User: 44 [279. 287.  28. 365. 121.  69. 122. 373. 305.  42.]
User: 45 [267. 356. 223.  93. 330. 191. 380. 471. 291. 367.]
User: 46 [296. 308.   4. 143. 295. 261. 392. 245. 272. 170.]
User: 47 [232. 263. 137. 259. 344. 242. 378. 108. 319. 398.]
User: 48 [492. 320. 458. 135. 325. 363.  80. 411. 222. 253.]
User: 49 [180. 204. 401.  38. 410. 412. 388. 471. 131. 223.]
```

(b)

Task 2

Για την άσκηση αυτή δημιουργήθηκαν 100 group των 5,10,15,20 χρηστών μέσω της συνάρτησης create_random_group()

Single Item Recommendation

- Borda Count

  Με την χρήση του Borda Count στόχος αποτελεί η πρόταση items που είναι ευραίως αποδεκτά από τους χρήστες του group. Λόγω της φύσης του μηχανισμού παρατηρείται ότι items τα οποία κάποιοι χρήστες αγαπούν ή μισούν δεν προτείνονται, μιας και είτε θα βρίσκονται πολύ ψηλά είτε πολύ χαμηλά στην κατάταξη. Από την άλλη, items τα οποία μπορεί να μην είναι αγαπημένα για πολλούς χρήστες, αλλά θεωρούνται σχετικά αποδεκτά προτείνονται μιας και έχουν μια συνεχή άυξηση των βαθμών τους.

  Έχοντας την ταξινομημένη prefList ως sortedPref, ο αλγορίθμος υλοποιήθηκε με την συνάρτηση borda_count(), η οποία για κάθε user του group, ανατρέχει κάθε item της ταξινομημένης λίστας του user και προσθέτει βαθμούς σε αυτό, ανάλογα με την θέση στην οποία το βρήκε. Τέλος, μέσω ταξινόμησης επιστρέφει το index του item που συγκέντρωσε τους περισσότερους βαθμούς για κάθε group.

  Τα αποτελέσματα του παραπάνω αλγορίθμου φαίνονται στον παρακάτω πίνακα:

Group Size	Average Rating
5	7.065
10	6.800
15	6.766
20	6.725

  Σημαντική παρατήρηση αποτελεί ότι επιβεβαιώνεται η ιδέα της πρότασης γενικά αποδεκτών items. Όπως θα ήταν φυσικό παρατηρείται επίσης ότι με την αύξηση του αριθμού των μελών του group, πρέπει να γίνουν συμβιβασμοί και έτσι το μέσο rating του item που προτείνεται φθίνει.

- Copeland Method

  Η μέθοδος Copeland αποτελεί εναλλακτική λύση για την επιλογή του βέλτιστου item για κάποιο group χρηστών. Η λογική της εν λόγω μεθόδου είναι να πραγματοποιηθεί σύγκριση για κάθε πιθανό ζεύγος αντικειμένων, αξιοποιώντας τις προτιμήσεις των χρηστών. Οι προτιμήσεις τους, όπως έχει αναφερθεί και παραπάνω, είναι αποθηκευμένες στην prefList.

  Αξιοποιώντας λοιπόν την prefList, εντοπίζεται για κάθε χρήστη ποιο item από τα δύο προτιμάει. Το item που ικανοποιεί κάθε χρήστη περισσότερο, κερδίζει έναν "προσωρινό πόντο" (roundWins). Στο τέλος και αφού έχουν πραγματοποιηθεί όλες οι συγκρίσεις, για όλους τους χρήστες, το item με

τους περισσότερους κερδισμένους "προσωρινούς πόντους", κερδίζει τελικά τον βαθμό. Στην περίπτωση που τα δύο αντικείμενα βγαίνουν ίσα στις προτιμήσεις των χρηστών, τότε θα μοιραστούν τον πόντο. Η διαδικασία αυτή επαναλαμβάνεται για κάθε πιθανό ζεύγος από items. Τέλος, με βάση τις παραπάνω μετρήσεις, επιλέγεται το αντικείμενο που έχει συγκεντρώσει τους περισσότερους βαθμούς, ως αυτό που ικανοποιεί το group.

Η μέθοδος Copeland είναι ορισμένη στην συνάρτηση copeland_method() και για τον γρηγοτότερο υπολογισμό όλων των πιθανών groups, έχει γίνει χρήση των threads, όπου το καθένα αναλαμβάνει τα ισάρηθμα groups. Τα αποτελέσματα που παράχθηκαν, παρουσιάζονται στον παρακάτω πίνακα

Group Size	Average Rating
5	7.053
10	6.925
15	6.759
20	6.743

Όπως και με το Borda Count, έτσι και με τη Copeland Method, παρατειρείται ότι όσο πιο μικρό είναι το μέγεθος του group, τόσο πιο πολύ θα ικανοποιεί τα μέλη του.

Multiple Item Recommendation

Για το Multiple Item Recommendation, εφαρμόσαμε τον μηχανισμό του Reweighted Approval Voting ο οποίος χρησιμοποιείται για να καθορίζει ένα committee από K νικητές μέσα σε μία ψηφοφορία. Το επιθυμητό χαρακτηριστικό αυτού του μηχανισμού είναι ότι, προτείνοντας περισσότερα από 1 αντικείμενα για μια ομάδα ανθρώπων, υπάρχει μεγαλύτερη ποικιλία και κατά συνέπεια πιθανότερη ευχαρίστηση των ατόμων που ψήφισαν. Ο μηχανισμός αποτελείται από K γύρους όπου σε κάθε γύρο εκλέγεται ένας υποψήφιος.

Σαν ορίσματα περνάμε στην συνάρτηση rav(Reweighted Approval Voting):

- groupIndexes: Ποιοι users βρίσκονται μέσα στο group
- prefList: Το preference του καθέ user για το κάθε item
- k : Το μέγεθος του committee των item που θα πρέπει να εκλεχθούν
- threshold : Το όριο του rating για το ποια items θεωρούνται αποδεκτά από κάποιον user. Στην περίπτωση μας είναι 6

Αρχικά, δημιουργείται το Approval List του κάθε user για να γνωρίζουμε εάν κάνει approve ένα item ή όχι ανάλογα με το threshold που έχουμε θέσει. Στην συνέχεια, για κάθε user που είναι μέσα στο group, παίρνουμε την λίστα με τα approved items που φτιάξαμε νωρίτερα και μετράμε πόσα items του user βρίσκονται μέσα

στο committee, για να υπολογίσουμε το βάρος με το οποίο θα πολλαπλασιάσουμε την ψήφο το. Ό τρόπος με τον οποίο υπολογίζεται είναι με τον παρακάτω τύπο:

$$\sum_{i:c\in A_i} \frac{1}{|S \cap A_i| + 1}$$

Δηλαδή, για τον κάθε user i αν το approved item Ai είναι μέσα στο committee θα προσθέτει +1 στον παρανομαστή του user i. Με αυτό τον τρόπο υπολογίζεται το βάρος που θα έχει ο κάθε user στις ψήφους του, σε οποιοδήποτε γύρο. Τέλος, υπολογίζουμε τον αριθμό των ψήφων συνυπολογίζοντας τα βάρη τα οποία φτιάξαμε και η μέγιστη τιμή, του weightedItemVotes είναι το item που εκλέξαμε. Συνεχίζεται η διαδικασία μέχρι να εκλεχθούν K items.

Task 3

Στο $3^ο$ μέρος της εργασίας χρειάστηκε να δημιουργηθούν groups ατόμων, όπου οι πεποιθήσεις τους για τα items είναι είτε παραπλήσιες, είτε διαφορετικές. Για τον παραπάνω σκοπό, αξιοποιήθηκε το μέτρο ομοιότητας Kendall T. Σύμφωνα με τον εν λόγω αλγόριθμο, υπολογίζεται το πλήθος των missmatch μεταξύ όλων των πιθανών συνδυασμών αντικειμένων για 2 χρήστες. Με αυτόν τρόπο υπολογίζεται το ποσοστό ομοιότητας/ανομοιότητας των 2 χρηστών.

Η λογική που ακολουθήθηκε προκειμένου να υλοποιηθεί η μέθοδος Kendall T, είναι η ακόλουθη: Αρχικά επιλέγεται τυχαία ένας χρήστης, ο οποίος θα αποτελεί και το βασικό μας παράγοντα, έτσι ώστε να τοποθετούνται κατάλληλα οι υπόλοιποι χρήστες. Για την δημιουργία των groups (5, 10, 15, 20), διαλέγονται με τυχαίο τρόπο και πάλι χρήστες, για να διαπιστωθεί εάν ταιριάζουν με τον αρχικό χρήστη. Ανάλογα με την ομοιότητά τους, η οποία ορίζεται με βάση ενός threshold, που έχει ορισθεί, κατατάσσεται στο κατάλληλο group. Στην περίπτωση, που δεν ικανοποιεί ούτε την συνθήκη ομοιότητας, αλλά ούτε την συνθήκη ανομοιότητας, παραλείπεται και επιλέγεται άλλος χρήστης. Η παραπάνω διαδικασία επαναλαμβάνεται έως ότου γεμίσει το group. Αξίζει να σημειωθεί, ότι για την παραπάνω υλοποίηση, αξιοποιήθηκαν process, προκειμένου τα αποτελέσματα να εξάγονται σε πιο "ρεαλιστικό" χρόνο.

Τέλος με βάση τα group που έχουν δημιουργηθεί, επαναλαμβάνεται το Task 2a, δηλαδή ο Borda Count και η Copeland Method. Τα αποτελέσματα παρουσιάζονται στους ακόλουθους πίνακες.

Αποτελέσματα Borda Count

Group Size	Average Rating (Similar)	Average Rating (Divergent)
5	7.685	6.956
10	7.404	7.182
15	7.345	7.399
20	6.885	6.944

Αποτελέσματα Copeland Method

Group Size	Average Rating (Similar)	Average Rating (Divergent)
5	7.655	7.013
10	7.404	7.267
15	7.328	7.406
20	6.869	6.9264

Μπορεί εύκολα να διαπιστωθεί, η βελτίωση του rating του item που επιλέγεται, για τα group που διαθέτουν παρόμοιους χρήστες. Το γεγονός αυτό είναι απόλυτα λογικό, καθώς οι προτιμήσεις τους ταυτίζονται σε μεγάλο βαθμό. Επίσης, παρατηρείται ότι ακόμα και αν οι χρήστες είναι διαφορετικοί μεταξύ τους, οι δύο μηχανισμοί επιλέγουν items που δεν τους ικανοποιούν τόσο, όσο στην περίπτωση του random task 2a, αλλά είναι σχετικά αποδεκτά από αυτού.

Task 4

Για την εκτέλεση του συγκεκριμένου Task φτιάχτηκε ένα random group 7 ατόμων (feasible περίπου 150 items)

a

Για την εύρεση των feasible items χρησιμοποιείται η συνάρτηση items_feasible(), στην οποία, αφού αθροίστηκαν τα budget των χρηστών του group, βρέθηκαν τα items που έχουν κόστος μικρότερο από το συνολικό budget του group.

b

Για την κατασκευή του μηχανισμού κατανομής κόστους του item, επιλέχθηκε ένα τυχαίο item από το set των feasible items του group. Η ιδέα του μηχανισμού είναι η εξής:

Ο εκάστοτε χρήστης $u \in U$ πρέπει να πληρώσει το ποσό $x_{u,i}$ για ένα item i μέσω της συνάρτησης:

$$x_{u,i} = \frac{r_{u,i} * cost_i}{\sum_{usr \in U} r_{usr,i}}$$

Στην περίπτωση που το budget δεν αρκεί για να πληρώσει το μερίδιό του, πληρώνει το budget του, ενώ το ποσό που δεν κατάφερε να πληρώσει μπαίνει σε ένα ταμείο διαμοιρασμού (sharedCost). Κάθε χρήστης που έχει υπολειπόμενο budget (δεν το έχει δώσει όλο) μπαίνει σε μία λίστα richUsers. Η παραπάνω διαδικασία αποτελεί τον πρώτο κύκλο διαμοιρασμού του κόστους. Όταν ολοκληρωθεί ο πρώτος γύρος διαμοιρασμού μπορεί να υπάρχει ένα ποσό του item που δεν έχει καλυφθεί ακόμα καθώς και μία λίστα richUsers. Σε αυτό τον γύρο οι richUsers θα μοιραστούν το ποσό του sharedCost, με την ίδια συνάρτηση, μόνο που αυτή την φορά το σύνολο $U = richUsers$ και $cost_i = sharedCost$. Το ποσό το οποίο θα αποφασίσει αυτός ο γύρος να πληρωθεί από κάθε έναν των εναπομεινάντων χρηστών προστίθεται στο ποσό που είχαν να πληρώσουν από τους προηγούμενους γύρους. Εάν κάποιος

χρήστης δεν μπορεί να καλύψει το νέο ποσό πληρωμής, τότε νέος γύρος με την ίδια διαδικασία με ακόμη μικρότερη λίστα richUsers έως ότου δεν υπάρχει πλέον sharedCost. Ο προαναφερθείς μηχανισμός εγγυάται ότι θα καταλήξει σε κάποιο payment vector με $sharedCost = 0$ μιας και το item επιλέγεται από ένα set feasible items που μπορούν οι χρήστες του group να αγοράσουν. Επίσης εγγυάται από την φύση του ότι κάθε χρήστης θα πληρώσει το πολύ το budget του.

Η παραπάνω ιδέα βασίζεται στο γεγονός ότι το group είναι συγκεκριμένο και σε καμία περίπτωση δεν θα μπορούσε να σπάσει ή να μειωθεί. Επίσης, θεωρεί ότι το item που έχει επιλεχθεί προς αγορά, δεν αλλάζει και ούτως ή άλλος θα αγοραστεί. Με αυτά τα δεδομένα, θεωρείται ότι οι πιο πλούσιοι χρήστες του group θα καλύψουν το κόστος του item για τα υπόλοιπα μέλη, ακόμα και αν δεν ταιριάζει και τόσο πολύ στους ίδιους. Βέβαια, αξίζει να αναφερθεί ότι ακόμα και αν κάποιος είναι πλούσιος, ο παραπάνω μηχανισμός εγγυάται ότι θα τον χρεώσει ανάλογα με το rating που έχει για το item, στην περίπτωση που υπάρχει κάποιος άλλος χρήστης με διαθέσιμα χρήματα και μεγαλύτερο rating για το item. Στην περίπτωση όμως που έχει μείνει μόνο ένας πλούσιος χρήστης ή πλούσιοι χρήστες με παρόμοια ratings, τότε αναγκαστικά θα μοιραστούν το υπολειπόμενο κόστος. Άλλωστε έτσι έκαναν και οι πλούσιοι Αθηναίοι με τα θεωρικά εισιτήρια του αρχαίου θεάτρου.

Task 5

Το metric το οποίο χρησιμοποιήσαμε για να ποσοτικοποιήσουμε το user satisfaction είναι με βάση τον παρακάτω τύπο:

$$sat_{u,i,x_u} = \alpha^{-\frac{r_{u,max}-r_{u,i}}{r_{u,max}}} \cdot \beta^{\frac{b_u-x_u}{b_u}}$$

Οι τιμές των παραμέτρων ισούται με α=8 και β=2 όπου το $r_{u,max}$ είναι το μέγιστο rating του user u, το $r_{u,i}$ είναι το rating για το item που του προτείνουμε, το b_u είναι το budget που διαθέτει, και x_u είναι το ποσό που θα πρέπει να πληρώσει, με τον payment mechanism που φτιάξαμε στο task 4. Στον κώδικά μας, το satisfaction υπολογίζεται στην συνάρτηση calculate_sat(), και χρησιμοποιεί την ανάλυση που κάναμε παραπάνω.

Δημιουργήσαμε 100 groups με μεγέθη g, όπου $g = \{4, 6, 8, 10, 12\}$ και υπολογίσαμε τον μέσο όρο satisfaction σύμφωνα με το παραπάνω metric. Τα αποτελέσματα φαίνονται παρακάτω:

Group Size	Average Satisfaction
4	0.341
6	0.656
8	1.029
10	1.395
12	1.740

Παρατηρούμε ότι όσο μεγαλώνει ο αριθμός των ατόμων του group τότε το Average Satisfaction ανεβάνει, πράγμα το οποίο μπορεί να οφείλεται στο ότι, όσο περισσότεροι είναι, τόσο μεγαλύτερο budget έχουν στην διάθεση τους και άρα

μεγαλύτερη ποικιλία στα item που μπορούν να αγοράσουν, που σημαίνει ότι είναι πιο πιθανό να πάρουν κάτι που τους αρέσει. Επιπλέον, με το να είναι περισσότεροι μέσα στο group, χρειάζεται να πληρώσουν λιγότερο για να αποκτήσουν το item που θα πάρουν. Αξίζει να σημειωθεί ότι η μέγιστη τιμή που μπορεί να πάρει το satisfaction είναι 2, ενώ η ελάχιστη είναι $\frac{1}{8} = 0.125$.