

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ

Διδάσκοντες: Σ. Λυκοθανάσης, Δ. Κουτσομητρόπουλος

Ακαδημαϊκό Έτος 2019-2020

Εργαστηριακή Άσκηση

Μέρος Β΄

Πετράκης Κωσταντίνος

ΑΜ:1041589(παλαιός:235878)

Έτος: 7^ο

Η υλοποίηση έγινε στο περιβάλλον Jupyter Notebook με χρήση της Python και της βιβλιοθήκης `ryeasga` για γενετικούς αλγορίθμους. Το αρχείο με τον κώδικα είναι:

<https://drive.google.com/drive/folders/1a5uW61ERgjECJ6ollmFO-srRQsXZWWIb?usp=sharing>

B1.Σχεδιασμός ΓΑ

α) Κωδικοποίηση

Επέλεξα να κωδικοποιήσω κάθε άτομο του πληθυσμού σαν ένα διάνυσμα 1682 θέσεων, όσες και οι ταινίες, το οποίο περιέχει τις αξιολογήσεις του χρήστη για κάθε ταινία. Κάθε άτομο δηλαδή θα αποτελείται από 1682 γονίδια, όπου κάθε γονίδιο θα αντιστοιχεί στην βαθμολογία του χρήστη για την εκάστοτε ταινία. Για τις ταινίες που ο χρήστης δεν έχει δώσει αξιολόγηση χρησιμοποιούμε την τιμή 0 για να δηλώσουμε την έλλειψη αξιολόγησης. Χρησιμοποιώντας την τιμή 0 όταν στην συνέχεια θα χρησιμοποιήσουμε την μετρική Pearson για να εξετάσουμε την ομοιότητα μεταξύ των χρηστών, θα ληφθούν υπ' όψιν μόνο οι αξιολογήσεις που έχουν δώσει ήδη οι χρήστες.

β) Πλεονάζουσες Τιμές

Με την κωδικοποίηση που επέλεξα δεν προκύπτουν πλεονάζουσες τιμές. Όλα τα άτομα του πληθυσμού θα είναι διανύσματα 1682 θέσεων με τιμές στο διάστημα $[0,5]$. Όπως θα δούμε στην συνέχεια και κατά την εκτέλεση του γενετικού αλγορίθμου έχω φροντίσει να μην καταστρατηγείται αυτό το διάστημα στο οποίο μπορούν να πάρουν τιμές οι αξιολογήσεις, δηλαδή το κάθε γονίδιο κάθε ατόμου του πληθυσμού θα έχει τιμή στο διάστημα $[0,5]$ καθόλη την εκτέλεση του αλγορίθμου.

γ) Αρχικός Πληθυσμός

Δημιουργώ τον αρχικό πληθυσμό διατηρώντας τα γονίδια που αντιστοιχούν στις βαθμολογίες που έχει δώσει ο επιλεγμένος χρήστης ως έχουν και για τα γονίδια που αντιστοιχούν σε ταινίες που δεν έχει αξιολογήσει ο χρήστης (δηλαδή έχουν τιμή 0) δίνω σαν τιμή την μέση τιμή της βαθμολογίας των χρηστών στην γειτονιά του χρήστη, σε αυτήν την ταινία. Να σημειώσω εδώ ότι αυτή είναι μια πολύ καλή αρχική εκτίμηση για την

βαθμολογία που θα έδινε ο χρήστης στην ταινία δεδομένου ότι οι χρήστες στην γειτονιά του είναι αυτοί με τα περισσότερα κοινά γούστα με αυτόν(έχουν αξιολογήσει σχεδόν με τον ίδιο τρόπο τις περισσότερες από τις ταινίες που έχει ήδη δει ο επιλεγμένος χρήστης). Για τα γονίδια που αντιστοιχούν σε ταινίες που δεν έχει αξιολογήσει ο επιλεγμένος χρήστης και δεν υπάρχει και αξιολόγηση από κάποιο χρήστη της γειτονιάς, επιλέγω τυχαία μια τιμή στο διάστημα [1,5]. Με αυτόν τον τρόπο οι αξιολογήσεις του χρήστη συμπεριλαμβάνονται στα γονίδια των ατόμων του πληθυσμού και θα παραμένουν σταθερές καθόλη την εκτέλεση του αλγορίθμου. Ο γενετικός θα αφορά στην τροποποίηση των τιμών των γονιδίων των ατόμων που αντιστοιχούν σε ταινίες που δεν έχει δει ο επιλεγμένος χρήστης. Τα γονίδια που αντιστοιχούν στις ταινίες που έχει αξιολογήσει δεν θα μεταβάλλονται. Ο αρχικός πληθυσμός δηλαδή θα αποτελείται από άτομα τα οποία έχουν όλα στα γονίδια που αντιστοιχούν στις ταινίες που έχει αξιολογήσει ο χρήστης τις τιμές των αξιολογήσεων του και στα υπόλοιπα γονίδια είτε τον μέσο όρο των χρήστων της γειτονιάς είτε μία τυχαία τιμή στο διάστημα [1,5].

Η συνάρτηση που αντιστοιχεί στην δημιουργία του αρχικού πληθυσμού είναι η `create_individual(data)` στον κώδικα.

δ) Διαδικασία Επιδιόρθωσης

Όπως εξηγήσα στην δημιουργία του αρχικού πληθυσμού οι τιμές των γονιδίων που αντιστοιχούν σε ταινίες που έχει αξιολογήσει ο επιλεγμένος χρήστης δεν θα μεταβάλλονται επομένως δεν είναι δυνατόν να προκύψουν μη νόμιμες λύσεις στην υλοποίηση μου. Όσον αφορά την επιδιόρθωση θα μπορούσαμε να κρατάμε σε ένα διάνυσμα τις αξιολογήσεις του χρήστη και τις θέσεις αυτών και κάθε φορά να αντικαθιστούμε αυτές τις θέσεις με τις νόμιμες λύσεις αλλά μοιάζει με περιττή δουλειά αν μπορούμε να εξασφαλίσουμε ότι δεν θα μεταβάλλονται οι υπάρχουσες αξιολογήσεις. Για την εφαρμογή ποινης θα μπορούσαμε να κρατάμε και πάλι σε ένα διάνυσμα τις αξιολογήσεις του χρήστη και να υπολογίζουμε την συσχέτιση Pearson της προκύπτουσας λύσης από το διάνυσμα των αρχικών αξιολογήσεων του χρήστη. Αυτή η τιμή θα αντιστοιχεί στο 'penalty' της πιθανής λύσης και θα αφαιρούνταν από το `fitness_score` της λύσης, δηλαδή από τον μέσο όρο της συσχέτισης Pearson της λύσης από την γειτονιά του χρήστη.

ε) Εύρεση γειτονιάς χρήστη

Ο υπολογισμός της γειτονιάς του χρήστη γίνεται με την συνάρτηση `find_neighborhood(user_id)` στον κώδικα. Η συνάρτηση δέχεται σαν όρισμα το `id` του χρήστη της επιλογής μας και επιστρέφει ένα numpy array με τα 10 διανύσματα αξιολογήσεων (άτομα) των χρηστών στην γειτονιά του. Πιο συγκεκριμένα βρίσκω την συσχέτιση Pearson του διανύσματος αξιολόγησης του επιλεγμένου χρηστή με όλους τους υπόλοιπους και κρατάω τα `id` των 10 χρηστών που έχουν την μεγαλύτερη συσχέτιση Pearson με τον επιλεγμένο user. Όσον αφορά τις μετρικές που προτείνονται η μετρική ομοιότητας συνιμητόνιου έχει το μειονέκτημα ότι δεν λαμβάνει υπ'οψιν τις όποιες διαφορές μπορεί να υπάρχουν στην κλίμακα με την οποία αξιολογούν ταινίες διαφορετικοί χρήστες. Για παράδειγμα κάποιος χρήστης μπορεί να χρησιμοποιεί γενικά υψηλές αξιολογήσεις 3-5 και κάποιος άλλος να βαθμολογεί πολύ αυστηρότερα στο διάστημα 1-3. Μια λύση σε αυτό θα ήταν να κανονικοποιήσουμε τις αξιολογήσεις στο διάστημα [0,1]. Όσον αφορά την

ευκλείδεια απόσταση μάλλον δεν είναι κατάλληλη για το πρόβλημα μας εδώ καθώς γνωρίζουμε ότι δεν έχει καλά αποτελέσματα όταν οι διαστάσεις αυξάνονται και τα δεδομένα είναι αραιά, που είναι 2 από τα κύρια χαρακτηριστικά του προβλήματος μας, έχουμε διανύσματα αξιολογήσεων 1682 διαστάσεων και τα περισσότερα είναι πολύ αραιά καθώς οι χρήστες δεν έχουν αξιολογήσεις αρκετές ταινίες. Ακόμη με την χρήση της ευκλείδειας απόστασης μικρές διαφορές αγνοούνται και μεγάλες διαφορές διογκώνονται (εξαιτίας τους τετραγωνισμού). Με βάση αυτό η ευκλείδεια απόσταση εδώ μπορεί να δώσει παραπλανητικά αποτελέσματα με βάση την απόσταση των διανυσμάτων σε λίγες μόνο μεμονωμένες μεταβλητές (πχ 2 διανύσματα με σχετικά όμοιες αξιολογήσεις που σε 3 θέσεις έχουν το ένα 0 και το άλλο 5 θα μας οδηγήσουν στο λανθασμένο συμπέρασμα ότι τα 2 διανύσματα απέχουν αρκετά). Και η απόσταση Manhattan παρουσιάζει λίγο πολύ τα ίδια προβλήματα με την ευκλείδεια απόσταση με την διαφορά ότι οι μικρές διαφορές δεν αγνοούνται και οι μεγάλες δεν διογκώνονται σε τόσο μεγάλο βαθμό. Από την άλλη μεριά η συσχέτιση Pearson είναι καταλληλότερη για δεδομένα πολλών διαστάσεων, και ανεξάρτητη από την κλίμακα που κυμαίνονται οι αξιολογήσεις των χρηστών που συγκρίνω. Για παράδειγμα αν ο χρήστης της επιλογής μας έχει σε 3 ταινίες τις βαθμολογίες 2/5, 1/5 και 2/5 και συγκρίνεται με κάποιον που στις ίδιες ταινίες έχει βαθμολογίες 4/5, 2/5 και 4/5 τότε αυτοί οι 2 χρήστες κρίνεται ότι έχουν ίδια γούστα στις ταινίες και θα τους προταθούν παρόμοιες ταινίες. Έχω αντικαταστήσει τις ελλειπείς τιμές με 0 για όλους τους χρήστες στο dataset όπου δεν έχουν αξιολόγηση για κάποια ταινία. Με αυτό τον τρόπο όταν επιλέγω έναν χρήστη στην γειτονία του θα βρεθούν χρήστες που έχουν όμοιες αξιολογήσεις με τον επιλεγμένο χρήστη, στις ταινίες που αυτός έχει αξιολογήσει.

στ) Συνάρτηση καταλληλότητας

Σαν συνάρτηση καταλληλότητας έχω χρησιμοποιήσει τον μέσο όρο της απόστασης Pearson του χρήστη από την γειτονία του χωρίς να έχω κανονικοποιήσει την μετρική, όπως φαίνεται και στην συνάρτηση fitness στον κώδικα. Κανονικοποιώντας την μετρική στο διάστημα [0,1] έπαιρνα παρόμοια αποτελέσματα με την μόνη διαφορά ότι εκεί που είχαμε πχ 0.21 μέση ομοιότητα από την γειτονία του παίρναμε $0.60 = (0.21 - (-1))/2$, δηλαδή το αποτέλεσμα που είχαμε και πριν απλά κανονικοποιημένο. Με τον τρόπο που έχει επιλεγεί η γειτονία του χρήστη (τα άτομα με την μέγιστη συσχέτιση Pearson με τις αξιολογήσεις αυτού) δεν θα υπάρχουν αρνητικές συσχετίσεις στην γειτονία. Ίσως σε κάποια πιο σύνθετη υλοποίηση του αλγορίθμου να μπορούσαμε να εκμεταλλευτούμε κάπως την αρνητική συσχέτιση μεταξύ 2 χρηστών (στις ταινίες που βαθμολογεί υψηλά ο ένας θα έπρεπε να έχει χαμηλές βαθμολογίες ο άλλος και αντίστροφα). Όσο μεγαλύτερη είναι η απόλυτη τιμή της συσχέτισης Pearson τόσο μεγαλύτερη και η ομοιότητα μεταξύ των ατόμων. Η μέγιστη τιμή που μπορεί να πάρει είναι 1.

ζ) Γενετικοί τελεστές

Έχω χρησιμοποιήσει διασταύρωση μονού σημείου κατά την οποία επιλέγεται τυχαία μια θέση μεταξύ του 1 και του 1681 για το σημείο του ατόμου που θα πραγματοποιηθεί αυτή. Επειδή όλα τα άτομα του αρχικού πληθυσμού έχουν στα γονίδια που αντιστοιχούν στις

αξιολογήσεις του χρήστη τις ίδιες τιμές όποια 2 άτομα και αν επιλεγούν κατά την διαδικασία της διασταύρωσης, τα άτομα που θα προκύψουν θα έχουν κι αυτά στις ίδιες θέσεις τις αξιολογήσεις του χρήστη. Με τον τρόπο που κατασκεύασα τον αρχικό πληθυσμό δηλαδή είναι εγγυημένο ότι τα γονίδια που αντιστοιχούν στις αξιολογήσεις του χρήστη δεν θα πειραχτούν κατά την διαδικασία της διασταύρωσης. Το 'γενετικό υλικό' που ανταλλάσσεται είναι οι εκτιμήσεις των αξιολογήσεων για τις ταινίες που δεν έχει δει ο επιλεγμένος χρήστης. Χρησιμοποιώντας διασταύρωση πολλαπλού σημείου θα έχουμε παρόμοια αποτελέσματα καθώς και πάλι οι αξιολογήσεις του επιλεγμένου χρήστη δεν θα μεταβάλλονται παρά μόνο οι αξιολογήσεις για τις ταινίες που δεν έχει δει.

Για την μετάλλαξη αν επιλεγεί γονίδιο που αντιστοιχεί σε ταινία που έχει αξιολογήσει ο επιλεγμένος χρήστης τότε αυτή δεν εκτελείται. Αν το γονίδιο που επιλέχθηκε αντιστοιχεί σε κάποια από τις ταινίες που δεν έχει αξιολογήσει ο χρήστης τότε αντικαθιστούμε την βαθμολογία με μία τυχαία τιμή στο διάστημα [1,5] αλλά διαφορετική από αυτήν που υπάρχει ήδη εκεί. Αν πχ το γονίδιο που επιλέχθηκε για μετάλλαξη είχε τιμή 3 τότε αντικαθίσταται με μία από τις τιμές 1,2,4 ή 5.

B2. Υλοποίηση ΓΑ

Για την υλοποίηση χρησιμοποιήθηκε η βιβλιοθήκη `pygame`. Η συγκεκριμένη βιβλιοθήκη μας δίνει την δυνατότητα να ορίσουμε τις συναρτήσεις δημιουργίας του αρχικού πληθυσμού, `create_individual(data)`, την συνάρτηση αξιολόγησης `fitness` και τις συναρτήσεις διασταύρωσης(`crossover`) και μετάλλαξης(`mutate`) αντίστοιχα.

Με την χρήση αυτής της βιβλιοθήκης μπορούμε μόνο να καθορίσουμε σε πόσες γενεές θα σταματήσει ο γενετικός. Δεν μπορούμε να καθορίσουμε κριτήρια με βάση το `fitness` της υποψήφια λύσης. Έτσι για όλες τις δοκιμές χρησιμοποίησα αριθμό γενεών 100 κυρίως λόγω χρόνου αλλά και εξαιτίας της εκτέλεσης του αλγορίθμου 10 φορές για κάθε περίπτωση. Ακόμη και με χρήση 1000 γενεών όμως δεν παρατηρήθηκε κάποια σημαντική βελτίωση.

Να σημειώσω ότι για το μέρος B χρησιμοποιήθηκε ο χρήστης με `user_id=1`. Ο συγκεκριμένος χρήστης έχει 272 αξιολογημένες ταινίες από τις 1682. Αν επιλεγεί κάποιος χρήστης με περισσότερες αξιολογημένες ταινίες όπως πχ ο χρήστης `user_id = 7` που έχει περισσότερες από 400 αξιολογημένες ταινίες οι μετρικές συσχέτισης θα είναι λίγο μεγαλύτερες γενικά αλλά οι διαφορές όσον αφορά τις πιθανότητες διασταύρωσης και μετάλλαξης και το μέγεθος του πληθυσμού θα είναι ίδιες. Η μετρική συσχέτισης προκύπτει μεγαλύτερη εξ'αίτίας του τρόπου που δημιουργώ τον αρχικό πληθυσμό. Όλα τα άτομα του πληθυσμού έχουν ίσες τιμές στα γονίδια που αντιστοιχούν στις ταινίες που έχει ήδη αξιολογήσει ο χρήστης. Επομένως όσες περισσότερες ταινίες έχει αξιολογήσει ο επιλεγμένος χρήστης τόσο πιο πολλά γονίδια θα είναι κοινά μεταξύ των ατόμων του πληθυσμού και η συσχέτιση `Pearson` μεταξύ τους θα είναι μεγαλύτερη.

B3. Αξιολόγηση και επίδραση των παραμέτρων

α)

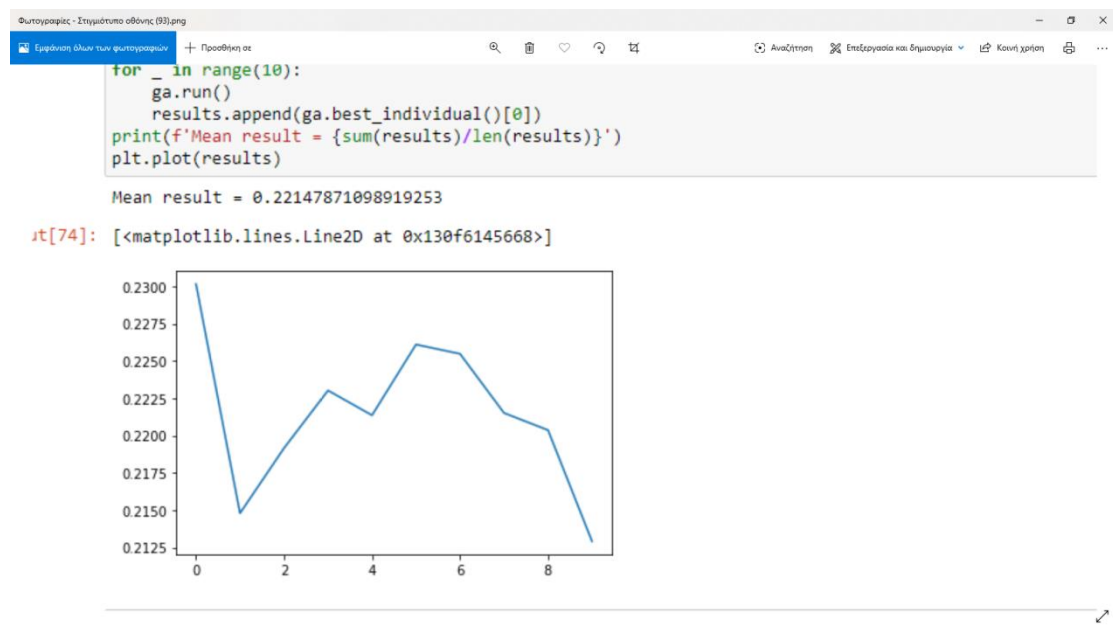
Εχουμε εκτελέσει όλα τα παρακάτω για το χρήστη με user id =1. Μπορείτε να χρησιμοποιήσετε οποιονδήποτε άλλο χρήστη απλά αλλάζοντας στον κώδικα την ανάθεση στην μεταβλητή user_id.

A/A	ΜΕΓΕΘΟΣ ΠΛΗΘΥΣΜΟΥ	ΠΙΘΑΝΟΤΗΤΑ ΔΙΑΣΤΑΥΡΩΣΗΣ	ΠΙΘΑΝΟΤΗΤΑ ΜΕΤΑΛΛΑΞΗΣ	ΜΕΣΗ ΤΙΜΗ ΒΕΛΤΙΣΤΟΥ	ΜΕΣΟΣ ΑΡΙΘΜΟΣ ΓΕΝΕΩΝ
1	20	0.6	0.00	0.2214	100
2	20	0.6	0.01	0.2172	100
3	20	0.6	0.10	0.2350	100
4	20	0.9	0.01	0.2252	100
5	20	0.1	0.01	0.2084	100
6	200	0.6	0.00	0.2449	100
7	200	0.6	0.01	0.2580	100
8	200	0.1	0.01	0.2401	100
9	200	0.9	0.01	0.2664	100

β) Σε όλα τα παρακάτω το μέσο αποτέλεσμα που παρουσιάζεται αφορά στον μέσο όρο της απόδοσης του καλύτερου ατόμου σε κάθε τρέξιμο.

Ακολουθούν στιγμιότυπα από την βέλτιστη λύση για κάθε ένα από τα 10 τρεξίματα του γενετικού για κάθε περίπτωση

pop_size = 20, $p_c = 0.6$ $p_m = 0.00$



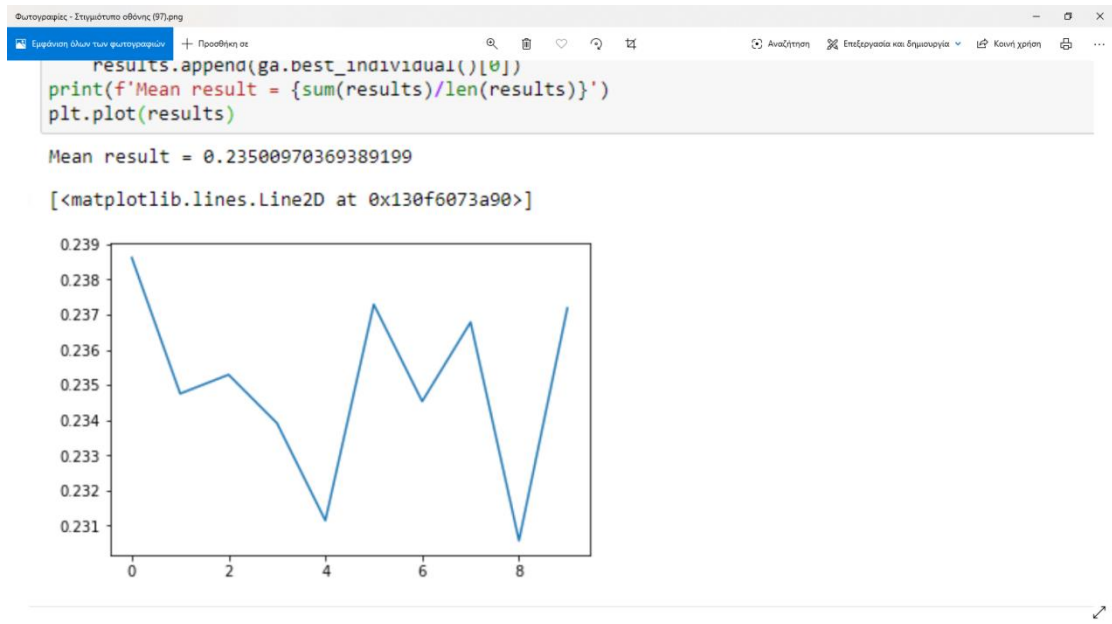
$pop_size = 20, p_c = 0.6, p_m = 0.01$



Ενώ παραθέτουμε και ένα παράδειγμα με τις ίδιες παραμέτρους και χρήση ελιτισμού



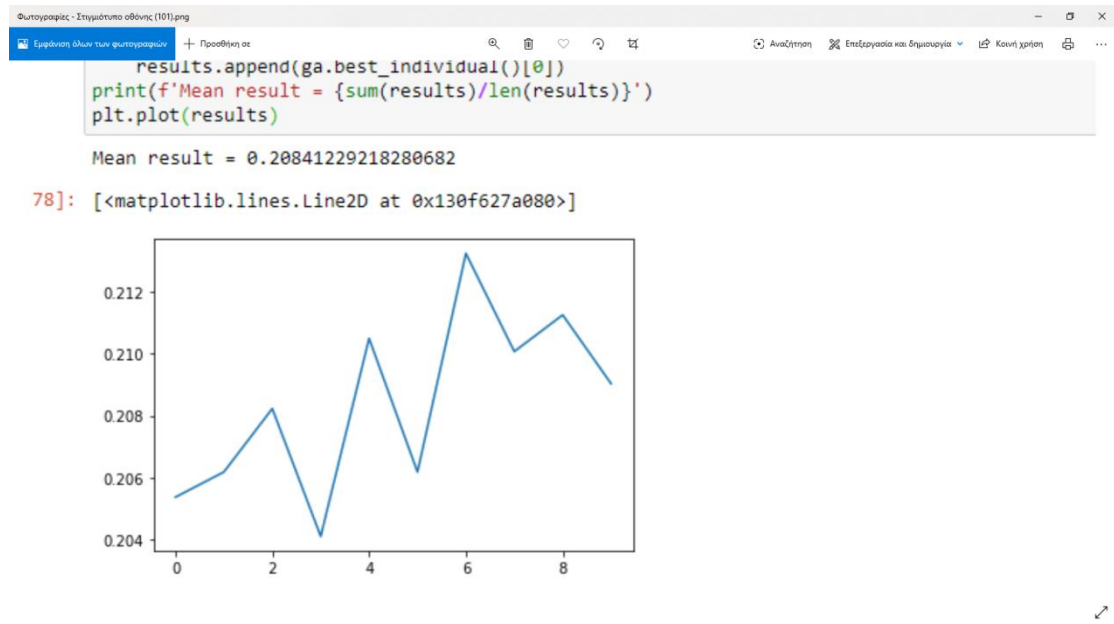
$pop_size = 20, p_c = 0.6, p_m = 0.10$



$pop_size = 20, p_c = 0.9, p_m = 0.01$



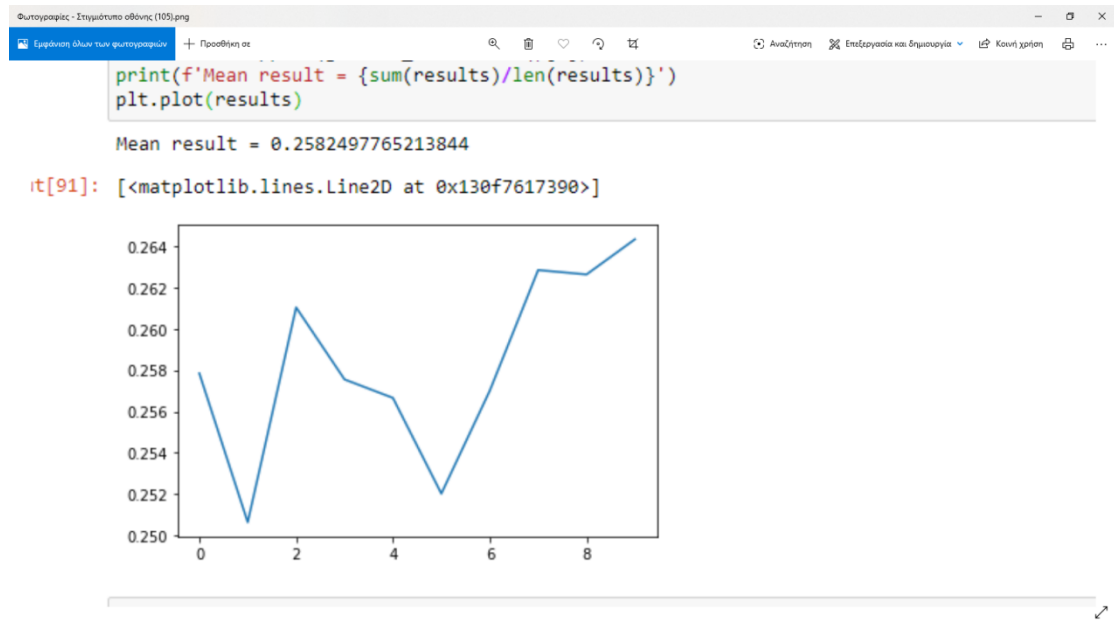
$pop_size = 20, p_c = 0.1, p_m = 0.01$



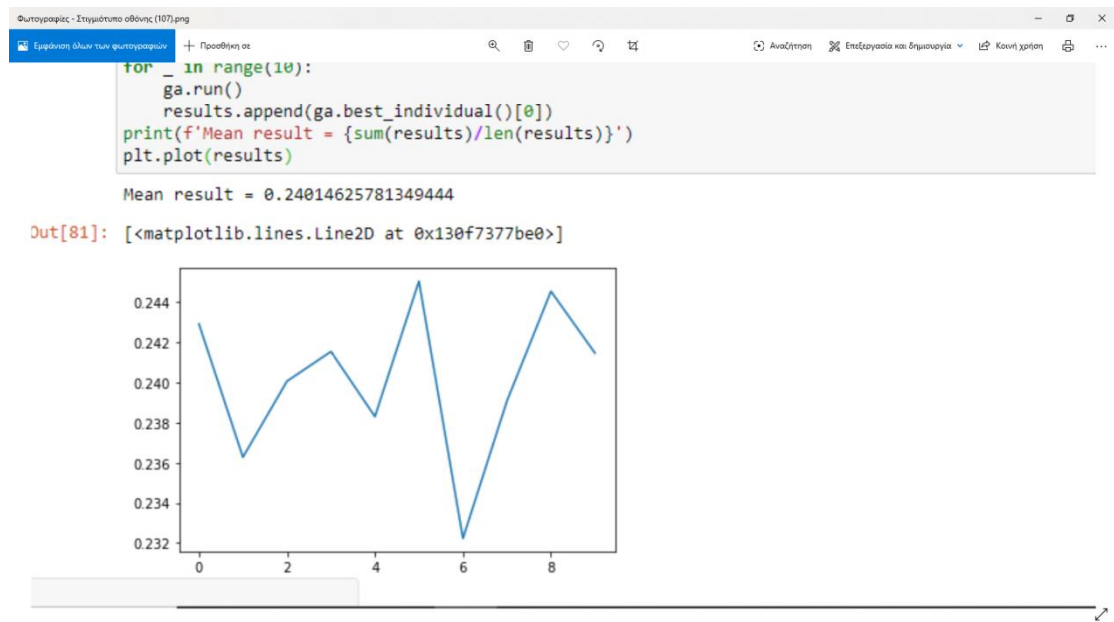
$pop_size = 200, p_c = 0.6, p_m = 0.00$



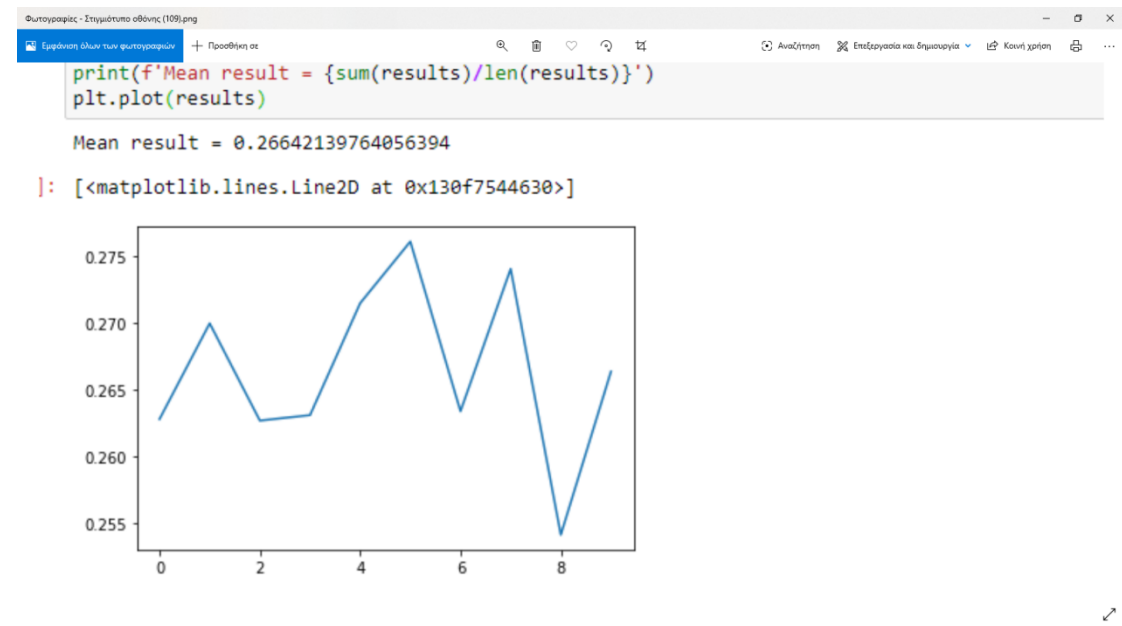
$\text{pop_size} = 200, p_c = 0.6, p_m = 0.01$



$\text{pop_size} = 200, p_c = 0.1, p_m = 0.01$



$pop_size = 200, p_c = 0.9, p_m = 0.01$



γ) Με μέγεθος πληθυσμού 20 άτομα βλέπουμε ότι πετυχαίνουμε τα καλύτερα αποτελέσματα με την μέγιστη πιθανότητα μετάλλαξης 0.1. Αυτό είναι αναμενόμενο σύμφωνα και με όσα γνωρίζουμε από την θεωρία καθώς περιοριζόμαστε από τον μικρό αριθμό πιθανών λύσεων που έχουμε με μόνο 20 άτομα αλλά όταν επιτρέψουμε μετάλλαξη στο 10% των γονιδίων του πληθυσμού ο γενετικός εξευρενά(exploration) μεγαλύτερο μέρος του χώρου των λύσεων και αποφεύγουμε τα άτομα να γίνουν ίδια από ένα σημείο και μετά. Από την άλλη παρατηρούμε ότι επιλέγοντας μόνο το 10% των ατόμων του πληθυσμού για διασταύρωση, δηλαδή εδώ $10\% * 20 = 2$ άτομα, έχουμε φτωχότερα αποτελέσματα, κι αυτό αναμενόμενο σύμφωνα με την θεωρία καθώς περιορίζουμε πολύ τον αριθμό των ατόμων που θα δημιουργήσουν απογόνους, άρα και πιθανές καλές λύσεις.

Με μέγεθος πληθυσμού 200 άτομα έχουμε γενικά καλύτερα αποτελέσματα σε σχέση με τις εκτελέσεις με μέγεθος πληθυσμού 20 άτομα. Αυτό γιατί έχουμε μεγαλύτερο πλήθος ατόμων που μπορεί να οδηγήσουν σε μια πιθανόν καλύτερη λύση, άρα και μεγαλύτερη ποικιλία πιθανών λύσεων. Τα καλύτερα αποτελέσματα τα έχουμε με πιθανότητα διασταύρωσης 0.9. Αυτό ίσως οφείλεται στο ότι υπάρχουν πολλά 'καλά' άτομα στον αρχικό πληθυσμό και αυξάνοντας την πιθανότητα διασταύρωσης επιτρέπουμε σε αυτά να συνεισφέρουν τα χαρακτηριστικά τους στην επόμενη γενιά. Το γεγονός ότι υπάρχουν πολλά καλά άτομα στον αρχικό πληθυσμό είναι αποτέλεσμα του τρόπου με τον οποίο φτιάχνω τον αρχικό πληθυσμό. Όπως εξηγήσα και παραπάνω κάθε άτομο του αρχικού πληθυσμού έχει τις αξιολογήσεις του επιλεγμένου χρήστη άθικτες, στα γονίδια που αντιστοιχούν σε ταινίες που δεν έχει αξιολογήσει ο χρήστης βάζουμε την μέση τιμή των αξιολογήσεων των χρηστών στην γειτονιά του επιλεγμένου χρήστη και όταν δεν έχουν αξιολογήσει κάποια ταινία ούτε οι χρήστες της γειτονιά τότε επιλέγουμε μια τυχαία τιμή στο διάστημα [1,5]. Συγκρίσιμα με

την καλύτερη περίπτωση αποτελέσματα έχουμε και με πιθανότητα διασταύρωσης 0.6. Επιπλέον βλέπουμε ότι με χαμηλή πιθανότητα διασταύρωσης 0.1, δηλαδή περιορίζοντας τα διαθέσιμα άτομα που θα δημιουργήσουν απογόνους, ακόμη και 200 άτομα αρχικό πληθυσμό δεν επιτρέπουμε στον γενετικό να φτάσει σε μια καλή λύση.

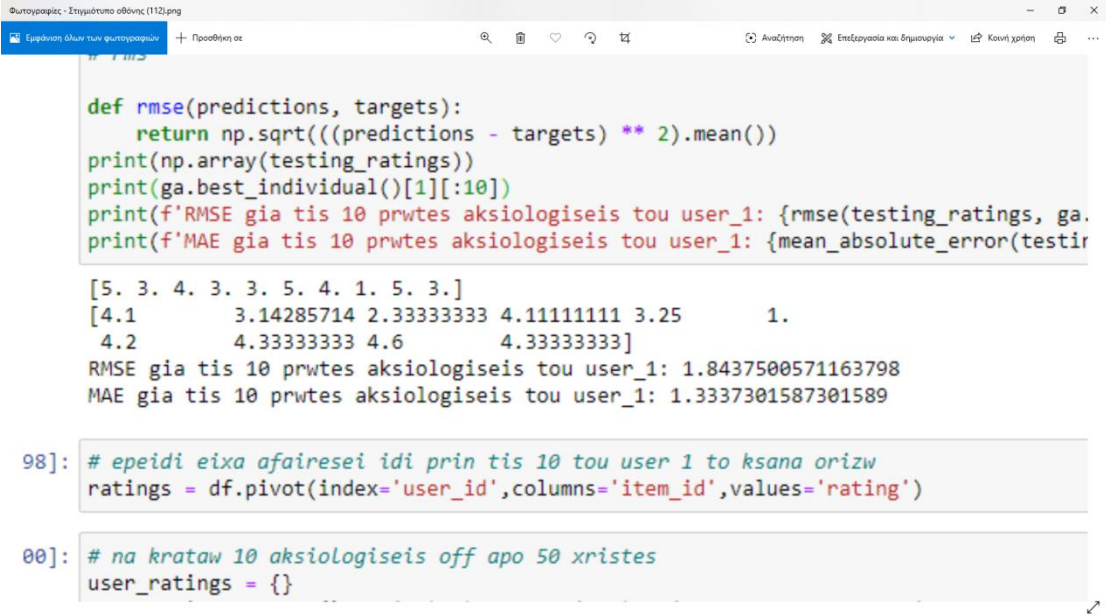
Τέλος βλέπουμε ότι η χρήση ελιτισμού στην περίπτωση όπου είχαμε μέγεθος πληθυσμού 20 άτομα με πιθανότητα διασταύρωσης $p_c = 0.6$ και πιθανότητα μετάλλαξης $p_m = 0.01$ έδωσε λίγο καλύτερα αποτελέσματα από ότι είχαμε χωρίς ελιτισμό. Πήραμε μέση συσχέτιση Pearson του καλύτερου ατόμου από την γειτονιά του 0.22 σε σχέση με 0.2172 που είχαμε χωρίς ελιτισμό.

Αξίζει να ανέφερω ότι πέτυχα καλύτερα αποτελέσματα από τα παραπάνω, δηλαδή μέση απόσταση από την γειτονιά σύμφωνα με την μετρική Pearson 0.30, με pop_size=200 πιθανότητα διασταύρωσης $p_c = 0.6$ και πιθανότητα μετάλλαξης $p_m = 0.1$. Αλλά επειδή αυτός ο συνδιασμός δεν δίνεται στον πίνακα απλά υπάρχει ένα cell στο notebook που δείχνει και το τρέξιμο του γενετικού με αυτές τις παραμέτρους.

B4. Αξιολόγηση Συστάσεων

α) Αφαιρώ τις 10 πρώτες αξιολογήσεις του χρήστη με user_id = 1 από το dataset και εκτελώ τον γενετικό αλγόριθμο με παραμέτρους pop_size = 200, $p_c = 0.9$ $p_m = 0.01$ για να ελέγξω πόσο κοντά θα είναι οι αξιολογήσεις που θα προβλέψει ο αλγόριθμος για αυτές τις 10 ταινίες με τις πραγματικές.

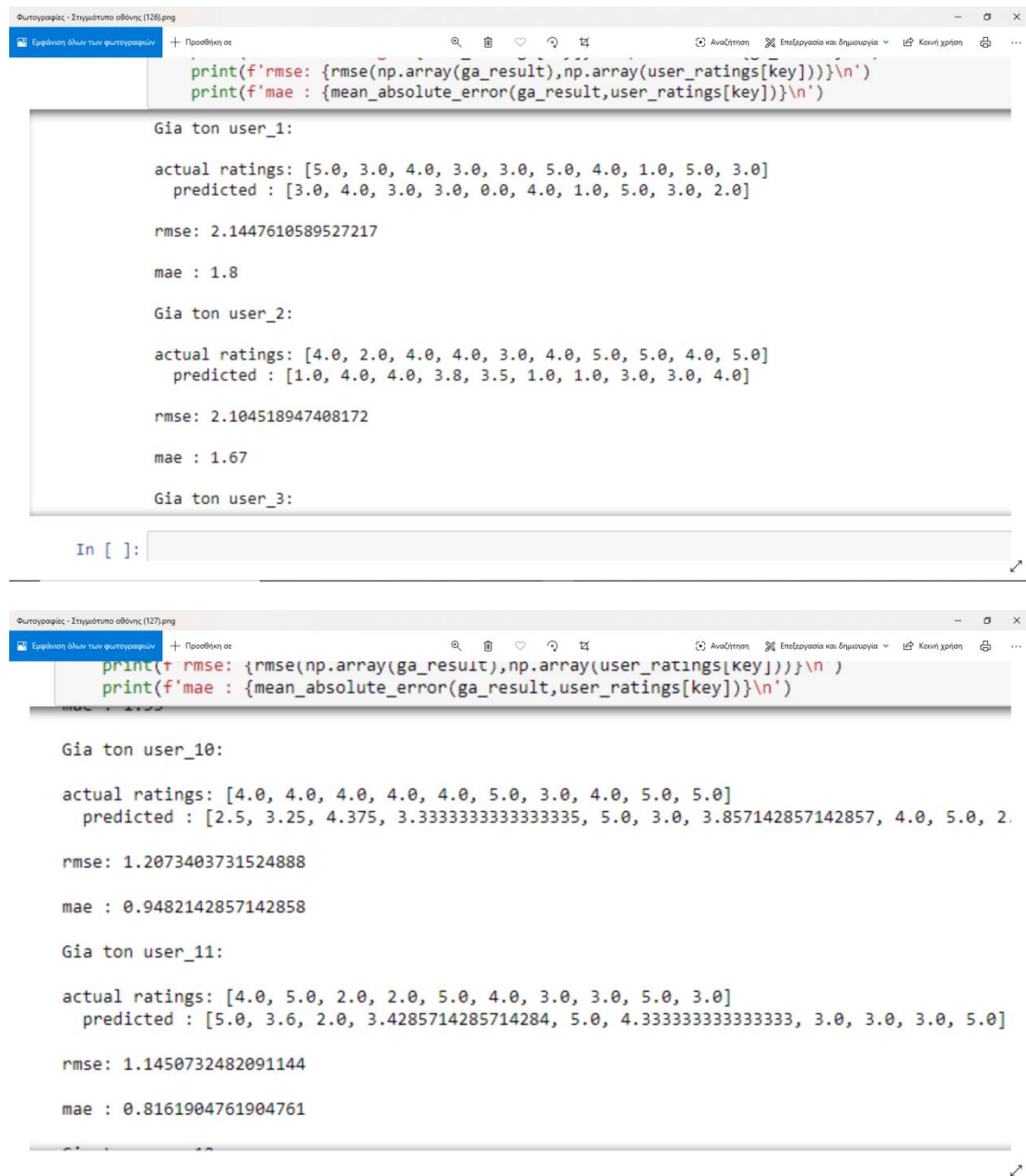
β) Στο στιγμιότυπο που ακολουθεί φαίνονται οι 10 πρώτες αξιολογήσεις του χρήστη 1, οι 10 αξιολογήσεις που προέβλεψε ο αλγόριθμος καθώς και τα RMSE και MAE.



```
def rmse(predictions, targets):  
    return np.sqrt(((predictions - targets) ** 2).mean())  
print(np.array(testing_ratings))  
print(ga.best_individual()[1][:10])  
print(f'RMSE gia tis 10 prwtes aksiologiseis tou user_1: {rmse(testing_ratings, ga.  
print(f'MAE gia tis 10 prwtes aksiologiseis tou user_1: {mean_absolute_error(testir  
  
[5. 3. 4. 3. 3. 5. 4. 1. 5. 3.]  
[4.1      3.14285714 2.33333333 4.11111111 3.25      1.  
4.2      4.33333333 4.6      4.33333333]  
RMSE gia tis 10 prwtes aksiologiseis tou user_1: 1.8437500571163798  
MAE gia tis 10 prwtes aksiologiseis tou user_1: 1.3337301587301589  
  
98]: # epeidi eixa afairesei idi prin tis 10 tou user 1 to ksana orizw  
ratings = df.pivot(index='user_id',columns='item_id',values='rating')  
  
00]: # na krataw 10 aksiologiseis off apo 50 xristes  
user_ratings = {}
```

Βλέπουμε ότι οι αξιολογήσεις που προέβλεψε ο γενετικός απέχουν κατά μέσο όρο 1.33 από τις πραγματικές σύμφωνα με τον μέγιστο απόλυτο σφάλμα.

γ) Εκτελώ την παραπάνω διαδικασία για τους 50 πρώτους χρήστες στο dataset . Παρακάτω φαίνονται ενδεικτικά κάποια στιγμιότυπα των αποτελεσμάτων. Οι μετρικές RMSE και MAE και για τους 50 χρήστες μπορούν να εξεταστούν με μεγαλύτερη λεπτομέρεια στο notebook. Παρατίθεται οι πραγματικές αξιολογήσεις του κάθε χρηστή , οι αξιολογήσεις που προέβλεψε ο γενετικός αλγόριθμος καθώς και το RMSE και το MAE.



The image shows two screenshots of a Jupyter Notebook interface. The top screenshot displays the output for three users (user_1, user_2, user_3). The bottom screenshot displays the output for two more users (user_10, user_11). Each user's output includes their actual ratings, predicted ratings, RMSE, and MAE.

```
print(f'rmse: {rmse(np.array(ga_result),np.array(user_ratings[key]))}\n')
print(f'mae : {mean_absolute_error(ga_result,user_ratings[key])}\n')
```

Gia ton user_1:

actual ratings: [5.0, 3.0, 4.0, 3.0, 3.0, 5.0, 4.0, 1.0, 5.0, 3.0]
predicted : [3.0, 4.0, 3.0, 3.0, 0.0, 4.0, 1.0, 5.0, 3.0, 2.0]

rmse: 2.1447610589527217

mae : 1.8

Gia ton user_2:

actual ratings: [4.0, 2.0, 4.0, 4.0, 3.0, 4.0, 5.0, 5.0, 4.0, 5.0]
predicted : [1.0, 4.0, 4.0, 3.8, 3.5, 1.0, 1.0, 3.0, 3.0, 4.0]

rmse: 2.104518947408172

mae : 1.67

Gia ton user_3:

In []:

```
print(f'rmse: {rmse(np.array(ga_result),np.array(user_ratings[key]))}\n')
print(f'mae : {mean_absolute_error(ga_result,user_ratings[key])}\n')
```

Gia ton user_10:

actual ratings: [4.0, 4.0, 4.0, 4.0, 4.0, 5.0, 3.0, 4.0, 5.0, 5.0]
predicted : [2.5, 3.25, 4.375, 3.3333333333333335, 5.0, 3.0, 3.857142857142857, 4.0, 5.0, 2.0]

rmse: 1.2073403731524888

mae : 0.9482142857142858

Gia ton user_11:

actual ratings: [4.0, 5.0, 2.0, 2.0, 5.0, 4.0, 3.0, 3.0, 5.0, 3.0]
predicted : [5.0, 3.6, 2.0, 3.4285714285714284, 5.0, 4.333333333333333, 3.0, 3.0, 3.0, 5.0]

rmse: 1.1450732482091144

mae : 0.8161904761904761

```
Φωτογραφίες - Στοιχείο οθόνης (128).png
Εμφάνιση όλων των φωτογραφιών Προσθήκη σε Αναζήτηση Επεξεργασία και δημοσίευση Κανόνες χρήσης ...

print(rmse: {rmse(np.array(ga_result),np.array(user_ratings[key]))})\n
print(f'mae : {mean_absolute_error(ga_result,user_ratings[key])}\n')

mae : 1.45

Gia ton user_14:

actual ratings: [5.0, 4.0, 5.0, 4.0, 3.0, 4.0, 3.0, 5.0, 3.0, 5.0]
predicted : [4.285714285714286, 4.0, 4.0, 3.0, 4.0, 1.0, 5.0, 2.666666666666665, 5.0, 1.0]

rmse: 2.0482833916740404

mae : 1.7047619047619047

Gia ton user_15:

actual ratings: [1.0, 1.0, 4.0, 1.0, 4.0, 4.0, 1.0, 3.0, 3.0, 5.0]
predicted : [1.0, 3.0, 2.6666666666666665, 4.0, 4.0, 2.0, 2.0, 2.0, 1.0, 3.0]

rmse: 1.6964014199999298

mae : 1.4333333333333333
```

```
Φωτογραφίες - Στοιχείο οθόνης (131).png
Εμφάνιση όλων των φωτογραφιών Προσθήκη σε Αναζήτηση Επεξεργασία και δημοσίευση Κανόνες χρήσης ...

print(rmse: {rmse(np.array(ga_result),np.array(user_ratings[key]))})\n
print(f'mae : {mean_absolute_error(ga_result,user_ratings[key])}\n')

mae : 0.845

Gia ton user_23:

actual ratings: [5.0, 4.0, 4.0, 4.0, 4.0, 4.0, 3.0, 3.0, 4.0, 4.0]
predicted : [3.2, 4.0, 4.4, 4.0, 3.25, 4.0, 2.5, 1.0, 1.0, 4.0]

rmse: 1.3119641763401926

mae : 0.845

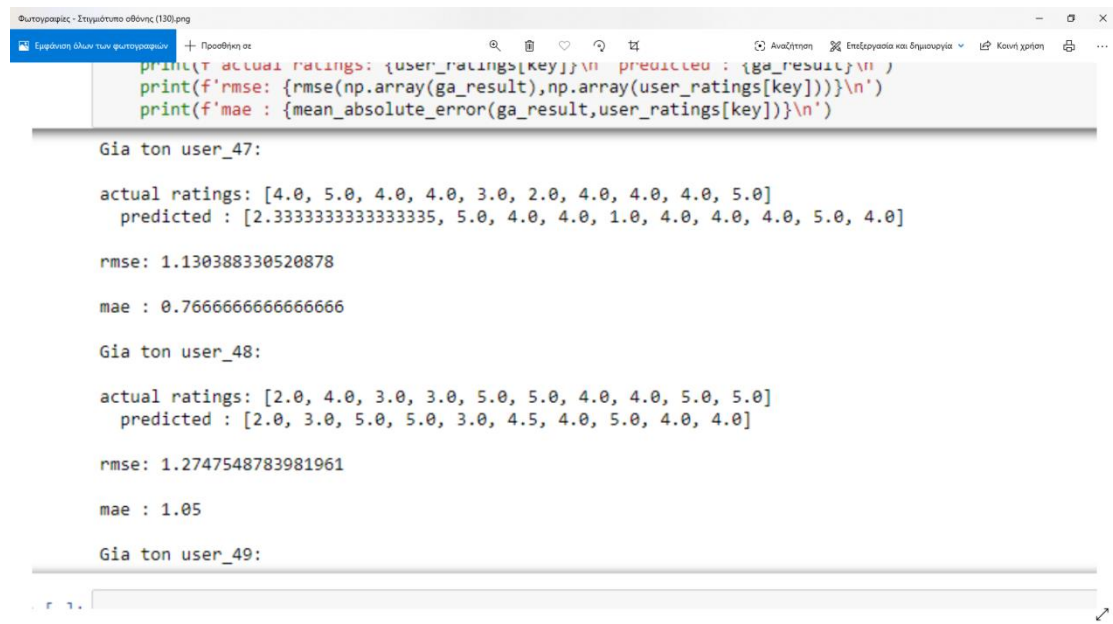
Gia ton user_24:

actual ratings: [4.0, 5.0, 5.0, 5.0, 5.0, 4.0, 5.0, 5.0, 4.0, 3.0]
predicted : [5.0, 5.0, 1.0, 5.0, 3.666666666666665, 1.0, 3.7142857142857144, 4.0, 4.5, 4.0]

rmse: 1.7799112057141382

mae : 1.3119047619047621

In [ ]:
```



```
print(f'actual ratings: {user_ratings[key]}\n predicted : {ga_result}\n')
print(f'rmse: {rmse(np.array(ga_result),np.array(user_ratings[key]))}\n')
print(f'mae : {mean_absolute_error(ga_result,user_ratings[key])}\n')

Gia ton user_47:

actual ratings: [4.0, 5.0, 4.0, 4.0, 3.0, 2.0, 4.0, 4.0, 4.0, 5.0]
predicted : [2.3333333333333335, 5.0, 4.0, 4.0, 1.0, 4.0, 4.0, 4.0, 5.0, 4.0]

rmse: 1.130388330520878

mae : 0.7666666666666666

Gia ton user_48:

actual ratings: [2.0, 4.0, 3.0, 3.0, 5.0, 5.0, 4.0, 4.0, 5.0, 5.0]
predicted : [2.0, 3.0, 5.0, 5.0, 3.0, 4.5, 4.0, 5.0, 4.0, 4.0]

rmse: 1.2747548783981961

mae : 1.05

Gia ton user_49:
```

Για κάποιους χρήστες όπως πχ για τον χρήστη με `user_id=23` ή γι' αυτόν με `user_id=48` βλέπουμε ότι τα αποτελέσματα είναι πολύ καλά. Πτυχαίνουμε MAE της τάξης του 0.8 και του 0.7 αντίστοιχα. Αυτό σημαίνει ότι οι αξιολογήσεις που προέβλεψε ο γενετικός απέχουν κατά μέσο όρο 0.8 στην μία περίπτωση και κατά 0.7 στην άλλη από τις πραγματικές αξιολογήσεις του χρήστη. Καθόλου άσχημα αν σκεφτούμε την σχετική απλότητα της υλοποίησης και τη ταχύτητα των υπολογισμών, και καταφέραμε για κάποιους χρήστες να είμαστε λιγότερο από 1 μονάδα κοντά στην πραγματική λύση. Το RMSE όπως είχαμε εξηγήσει και στο πρώτο μέρος της εργασίας βαραίνει περισσότερο μεμονωμένα υψηλά λάθη. Αν επομένως σε ένα γονίδιο μιας πρόβλεψης απέχουμε αρκετά από την πραγματική αξιολόγηση του χρήστη αυτό θα μας δώσει υψηλό RMSE ακόμη κι αν στα υπόλοιπα γονίδια είμαστε πολύ κοντά στην πραγματική αξιολόγηση. Γι' αυτό και αποτελεί μάλλον καλύτερο δείκτη για το πρόβλημά μας εδώ το μέσο απόλυτο σφάλμα που μας δείχνει πόσο απέχουμε κατά μέσο όρο από τις πραγματικές αξιολογήσεις. Για τους περισσότερους χρήστες οι προβλέψεις του γενετικού απέχουν κατά μέσο όρο περίπου 1 με 1,5 μονάδα από τις πραγματικές αξιολογήσεις ενώ μόνο σε ελάχιστες μεμονωμένες περιπτώσεις οι προβλέψεις απέχουν κατά μέσο όρο 2 μονάδες από τις πραγματικές τιμές. Οι περιπτώσεις όπου το μέσο απόλυτο σφάλμα είναι κοντά στην μονάδα είναι πολύ περισσότερες από αυτές που πλησιάζει το 2.