ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Συνεργατικό Φιλτράρισμα Βασισμένο στους Χρήστες με την βοήθεια Γενετικών Αλγορίθμων User-Based Collaborative Filtering with the help of Genetic Algorithms
Ονοματεπώνυμο Φοιτητή	Τζιούνης Σταύρος
Πατρώνυμο	Αλέξανδρος
Αριθμός Μητρώου	Π19169
Επιβλέπων Καθηγητής	Σωτηρόπουλος Διονύσιος,Επίκουρος Καθηγητής

Ημερομηνία Παράδοσης: Φεβρουάριος 2025

Ευχαριστίες

Πρωτίστως, ευχαριστώ τον Καθηγητή κ.Σωτηρόπουλο Διονύση, για την συνεχή υποστήριξη και συμβολή του (καθώς και την υπομονή του!), στην ανάπτυξη της παρούσας προπτυχιακής διατριβής.

Στην συνέχεια, θέλω να ευχαριστήσω την αδερφή μου Αφροδίτη για την βοήθεια της κατά την μορφοποίηση του παρακάτω ακαδημαϊκου κειμένου.

Περίληψη

Τα συστήματα συστάσεων, αποτελούν έναν σημαντικό κλάδο της τεχνητής νοημοσύνης και αξιοποιούν διάφορες τεχνικές για να καταλήγουν σε έγκυρες προτάσεις στους χρήστες του συστήματος. Μια κατηγορία αλγορίθμων της Τεχνητής Νοημοσύνης, είναι οι Γενετικοί Αλγόριθμοι οι οποίοι εφαρμόζονται σε προβλήματα αναζήτησης και βελτιστοποίησης, όπου ο χώρος αναζήτησης λύσεων είναι μεγάλος και πολυπαραγοντικός. Σε αυτή την πτυχιακή αναπτύχθηκαν δύο τέτοια μοντέλα , ο GenRec1 και ο GenRec0 , που αναζητούν βέλτιστες διασυνδέσεις μεταξύ χρηστών, βοηθώντας στην πρόβλεψη των βαθμολογιών των χρηστών αυτών πάνω σε αντικείμενα. Τα μοντέλα αυτά αναλύονται ώς προς την μεθοδολογία τους καθώς και την προγραμματιστική τους υλοποίηση και αξιολογούνται τα αποτελέσματα τους σε σύγκριση με αντίστοιχες μεθόδους.

Λέξεις Κλειδιά:

Συστήματα Συστάσεων, Συνεργατικό Φιλτράρισμα, Γενετικοί Αλγόριθμοι, Τεχνητή Νοημοσύνη, Παραγοντοποίηση Πινάκων, Ημί-εποπτευόμενη Μάθηση, Χαμηλόβαθμες Υπερπαραμέτροι Πινάκων

Abstract

Recommender systems are an important subfield of Artificial Intelligence, making use of varied techniques, in order to come up with useful recommendations for the users of the RS. Genetic Algorithms are a category of algorithms used in order to solve Artificial Intelligence problems, where the search space for the solution is large and hard to search.In this undergraduate thesis two such algorithms where created, GenRec1 and GenRec0. They search for optimal connections between users, helping predict the rating these users give on certain items. The two models are analyzed on the basis of their methodology, their implementation in terms of code, as well as their results, in comparison to other common methods in the field.

KeyWords:

Recommender Systems, Collaborative Filtering, Genetic Algorithms, Artificial Intelligence, Matrix Factorization, Semi-supervised Learning,Low-rank Embeddings

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Διαγραμμάτων

Εισαγωγή στα Συστήματα Συστάσεων

Συστήματα Συστάσεων

Με τον όρο Συστήματα Συστάσεων (γνωστά και ώς **recommender systems**), περιγράφεται ένα σύνολο εξελιγμένων αλγορίθμων και τεχνικών που σχεδιάζουν και προτείνουν σε χρήστες μιας εφαρμογής ή υπηρεσίας, αντικείμενα σχετικά με τα ενδιαφέροντα και τις συμπεριφορές τους. Τα RS (Συστήματα Συστάσεων) άρχισαν να εξελίσσονται και να μοιάζουν με την σύγχρονη εκδοχή τους την δεκαετία του 1990, όταν για πρώτη φορά εφαρμόστηκε η τεχνική του Συνεργατικού Φιλτραρίσματος (Collaborative Filtering) σε ένα πειραματικό σύστημα ηλεκτρονικής αλληλογραφίας που ονομάστηκε Tapestry[1].

Σήμερα, η χρήση τους είναι διαδεδομένη σε πάρα πολλές πλατφόρμες και κοινωνικά δίκτυα του σύγχρονου διαδικτύου (Netflix, Spotify, Instagram), καθώς και σε πολλαπλούς ιστότοπους ηλεκτρονικού εμπορίου (Amazon, Skroutz, Temu) αποτελώντας βασικά εργαλεία για την προώθηση προϊόντων και άλλων αντικειμένων. Συγκεκριμένα, το μέγεθος της αγοράς συστημάτων συστάσεων ήταν γύρω στα 2.2 δισεκατομμύρια δολάρια για το έτος 2024 και προμηνύεται σταθερή αύξηση τουλάχιστον 10% άνα έτος, για το επόμενο διάστημα[2].

Ως ερευνητικό πεδίο τώρα, τα συστήματα συστάσεων εμφανίζουν αντίστοιχη πρόοδο, αποτελώντας έναν σημαντικό κλάδο της Τεχνητής Νοημοσύνης στον οποίο εφαρμόζονται πολλαπλές τεχνικές για την αξιοποίηση δεδομένων και την δημιουργία προτάσεων σε χρήστες. Διαφορετικές μέθοδοι βασισμένες στα μαθηματικά και στην Μηχανική Μάθηση έχουν προταθεί και αξιολογηθεί για την δημιουργία βέλτιστων συστάσεων αξιοποιόντας ποικιλία δεδομένων.

Αντικείμενο της Εργασίας αυτής

Σε αυτή την πτυχιακή διατριβή, θα ασχοληθώ με την επεξήγηση και την ανάλυση συγκεκριμένων αλγορίθμων που χρησιμοποιούνται σε αυτό το ερευνητικό πεδίο. Στην συνέχεια θα σχεδιάσω, θα αναλύσω και έπειτα θα υλοποιήσω δύο τέτοια μοντέλα. Οι αλγόριθμοι που έχω αναπτύξει ανήκουν στην υποκατηγορία αλγορίθμων συστάσεων που χρησιμοποιούν την μέθοδο Συνεργατικού Φιλτραρίσματος βασισμένη στους Χρήστες, συγκεκριμένα χρησιμοποιώντας δεδομένα από τις προτιμήσεις χρηστών πραγματοποιείτε μια διασύνδεση των χρηστών αυτών με βάση την ομοιότητα τους στα ενδιαφέροντα.

Έπειτα, μέσω της δημιουργίας και εκτέλεσης Γενετικού Αλγορίθμου, ο οποίος χρησιμεύει ώς εργαλείο βελτιστοποίησης του τρόπου διασύνδεσης χρηστών μεταξύ τους, καταλήγουμε σε ένα σύνολο προβλέψεων για το πώς θεωρούμε ο κάθε χρήστης θα

βαθμολογούσε το κάθε αντικείμενο. Στη συνέχεια, η συνολική διαδικασία εκτελείται για διαφορετικά σύνολα δεδομένων και διαφορετικούς πίνακες διασύνδεσης χρηστών, παρουσιάζοντας και αξιολογώντας τα αποτελέσματα κάθε φορά. Τα μοντέλα που αναπτύχθηκαν (GenRec1, GenRec2) κάνουν χρήση διαφόρων μαθηματικών τεχνικών με πρωτότυπο τρόπο, λύνοντας το πρόβλημα της βελτιστοποίησης πινάκων εγγύτητας μεταξύ χρηστών. Περιγράφεται αναλυτικά η υλοποίηση τους και τα αποτελέσματα αξιολογούνται και συγκρίνονται με άλλες μεθόδους. Προκύπτουν συμπεράσματα για την χρήση γενετικών αλγορίθμων στο πεδίο των Συστημάτων Συστάσεων εξετάζοντας τα θετικά και τα αρνητικά αυτής της προσέγγισης.

Διάρθρωση αυτής της εργασίας

Στο **Κεφάλαιο 1,** παρατίθεται το θεωρητικό υπόβαθρο της εργασίας αυτής, δίνοντας έμφαση στις διαφορετικές αλγοριθμικές μεθόδους που θα χρησιμοποιηθούν στην συνέχεια. Παρουσιάζονται τα Συστήματα Συστάσεων (RS) ώς ερευνητικό πεδίο μέσα από μία σύντομη ιστορική αναδρομή της εξέλιξης και της χρήσης τους. Περιγράφονται οι διάφορες κατηγορίες των RS μαζί με τα πλεονεκτήματα και μειονεκτήματα τους συμπεριλαμβάνοντας κάποια αντιπροσωπευτικά παραδείγματα για την κάθε κατηγορία. Γίνεται ιδιαίτερη εστίαση στην τεχνική Συνεργατικού Φιλτραρίσματος (Collaborative Filtering) και σε διάφορες μορφές της που έχει εμφανιστεί στην βιβλιογραφία. Έπειτα, γίνεται μια σύντομη παρουσίαση της Ημι-εποπτευόμενης Μάθησης, με επίκεντρο την διαχείριση συνόλων δεδομένων ώς επισημασμένα και μή επισημασμένα δεδομένα. Τέλος, αναλυεται η θεωρία πίσω από τους Γενετικους Αλγόριθμους

Στο **Κεφάλαιο 2,** αναλύεται η μεθοδολογία που ακολουθείται προγραμματιστικά σε επόμενα κεφάλαια. Παρουσιάζονται προηγούμενες προσπάθειες στην βιβλιογραφία, που είτε προσπαθούν να λύσουν το ίδιο πρόβλημα τεχνητής νοημοσύνης, είτε χρησιμοποιούν παρόμοιες τεχνικές με αυτές που χρησιμοποιούνται στην παρούσα εργασία. Στη συνέχεια, περιγράφεται η αναλυτικός σχεδιασμός των μοντέλων, που θα αναλυθούν συμπεριλαμβανομένων των αλγοριθμικών βημάτων, καθώς και του μαθηματικού υποβάθρου που σχετίζεται με αυτά. Στο τέλος του κεφαλαίου, αναφέρονται όλες οι τεχνολογίες που χρησιμοποιήθηκαν και ποιός είναι ο σκοπός και η χρήση της κάθε μίας, ξεχωριστά.

Στο **Κεφάλαιο 3,** παρουσιάζεται το σύνολο των δεδομένων που χρησιμοποιήθηκαν. Πραγματοποιείται καθαρισμός και προετοιμασία των δεδομένων και δημιουργούνται βοηθητικές συναρτήσεις για να συμβάλλουν στην εκμετάλλευση και αποδοτική χρήση τους. Στη συνέχεια, παρατίθενται τα αναλυτικά βήματα για την δημιουργία των μοντέλων GenRec1 και GenRec0 συμπεριλαμβανομένων σημαντικών χωρίων κώδικα που χρησιμοποιούνται κατα την εκτέλεση τους.

Στο **Κεφάλαιο 4,** πραγματοποιείται ανάλυση των αποτελεσμάτων από την εκτέλεση των μοντέλων χρησιμοποιώντας διαγράμματα και γραφικά στοιχεία καθώς και η σύγκριση με άλλες μεθόδους.

Τέλος, στο **Κεφάλαιο 5,** εστιάζω στα συμπεράσματα που προκύπτουν για την μορφή και την απόδοση των μοντέλων που ανέπτυξα, παραθέτοντας πιθανές βελτιώσεις και μελλοντικές κατευθύνσεις.

Κεφάλαιο 1: Θεωρητικό Υπόβαθρο

1.1: Συστήματα Συστάσεων

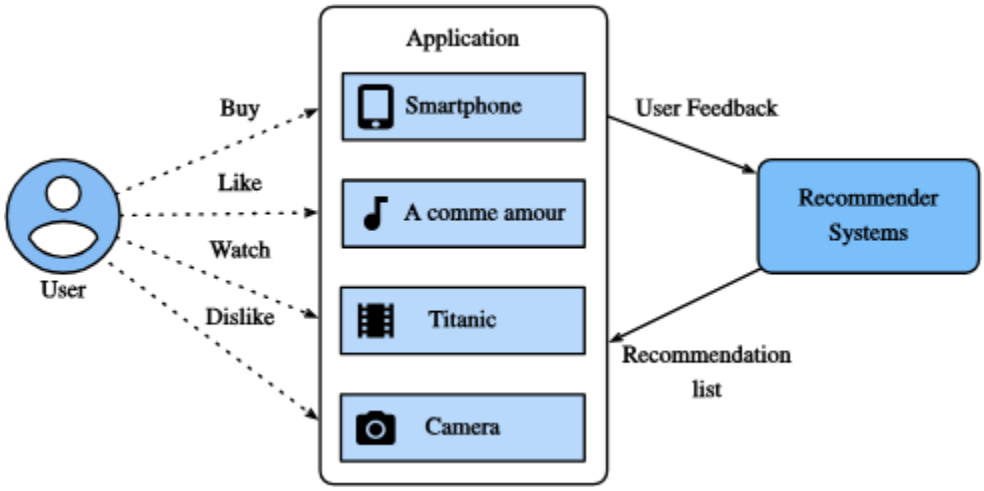
1.1.1: Σύντομη Ιστορική Αναδρομή

Τα Συστήματα συστάσεων δεν αποτελούν μια νέα εξέλιξη στον χώρο της τεχνητής νοημοσύνης και της πληροφορικής ,γενικότερα. Η ιδέα της δημιουργίας ενός συστήματος με το οποίο οι χρήστες θα μπορούν να δέχονται συστάσεις για αντικείμενα παρόμοια με αυτά που γνωρίζουν ήδη, πρωτοεμφανίστηκε το **1979** με το σύστημα Grundy, το πρώτο γνωστό σε εμάς Σύστημα Συστάσεων. Δημιουργήθηκε από την Αμερικανίδα επιστήμονα πληροφορικής Ελέιν Ριτς (Elaine Rich) και λειτουργούσε σαν έναν ψηφιακό βιβλιοθηκάριο ρωτώντας χρήστες συγκεκριμένες ερωτήσεις για τα ενδιαφεροντα τους με βάση τις απαντήσεις τους πρότεινε αντίστοιχα βιβλία από μια βάση δεδομένων.

Στην αρχή της δεκαετίας του 1990 εμφανίστηκαν δάφορα σημαντικά RS. Το **1992**, οι Belkin και Croft ανέπτυξαν την τεχνολογία ανάκτησης πληροφοριών ως θεμελιώδη για τις μηχανές αναζήτησης και το φιλτράρισμα πληροφοριών ως βάση για τα συστήματα συστάσεων. Όμως, το **1992** εμφανίστηκε το πρώτο σύστημα πληροφοριών συνεργατικού φιλτραρίσματος μέσω ανθρώπινης αξιολόγησης, το σύστημα Tapestry του Goldberg. Το σύστημα αυτο βασίστηκε σε μία μικρή κοινωνία , απο τις γνώμες των ανθρώπων της, οι οποίοι έγραφαν γράμματα για τις συμπεριφορές και απόψεις των υπολοίπων της . Τα κύρια συστατικά του αποτελούν το Indexer, Document Store. Annotation store, Filterer, Little box, Rrmailer, Appraiser, and Reader/Browser. Επιπλέον, το **1994**, παρουσιάστηκε το GroupLens σύστημα φιλτραρίσματος, το οποίο αφορούσε ειδήσεις και αυτομματοποιεί τη διαδικασία συνεργατικού φιλτραρίσματος βάσει κανόνων του Tapestry . Το GroupLens λειτούργησε με αξιολογήσεις που παρείχε μια κοινότητα χρηστών και χρησιμοποίησε τη μηχανική μάθηση για να προβλέψει εάν ένας χρήστης θα ήθελε συγκεκριμένα κρυφά μηνύματα. Τα βασικά του στοιχεία περιελάμβαναν WWW Server, Dynamic HTML Generator και ένα Σύστημα Συστάσεων . Στα τέλη του 1990 τώρα, συστήματα συστάσεων στο ηλεκτρονικό εμπόριο σημείωσαν επιτυχία, με το Amazon.com να είναι ένα από τα πρώτα που υιοθέτησαν την τεχνολογία συστάσεων σε μεγάλη κλίμακα .

Στην δεκαετία του 2000, εμφανίστηκαν επίσης διάφορα συστήματα συστάσεων. Το **2007** πραγματοποιήθηκε το πρώτο συνέδριο ACM Recommender Systems στο UMN. Αυτό το συνέδριο έχει γίνει ένα σημαντικό ετήσιο ακαδημαϊκό γεγονός, από τότε. Επιπλέον, την ίδια χρονιά, Οι Richardson κ.ά. παρουσίασαν ένα μοντέλο λογιστικής παλινδρόμησης (LR) που μείωσε σημαντικά το σφάλμα στην εκτίμηση του click-through rate . Το **2010**, ο Rendle πρότεινε τα Factorization Machines (FM), συνδυάζοντας τα πλεονεκτήματα των Support Vector Machines (SVM) και των factorization μοντέλων. Η

δεκαετία του 2010 έφερε με την σειρά της, πρωτοποριακές εξελίξεις στον τομέα αυτό. Τα Deep neural δίκτυα άρχισαν να χρησιμοποιούνται σε μοντέλα συστάσεων. Τα μοντέλα Wide&Deep και DeepFM αναπτύχθηκαν για τη βελτίωση των συστάσεων εφαρμογών. Τα YouTubeDNN και correct-sfx χρησιμοποιήθηκαν για την αύξηση της ακρίβειας των συστάσεων βίντεο. Τα DIN και DIEN προτάθηκαν για τη μοντελοποίηση διαδοχικών πληροφοριών, όπως τα ενδιαφέροντα των χρηστών με μηχανισμούς προσοχής. Τα DCN και DCN V2 αναπτύχθηκαν για να μάθουν αυτόματα και αποτελεσματικά τις προγνωστικές bounded-degree χαρακτηριστικές αλληλεπιδράσεις. Προτάθηκαν επίσης σημαντικά βαθιά μοντέλα συστάσεων, όπως τα FNN, PNN, NeuralCF, NFM και CVAE1 . Το **2018** ο Thorsten δίδασκε ένα μάθημα με τίτλο Counterfactual Machine Learning . Σήμερα, υπάρχουν όλο και περισσότερες μελέτες σχετικά με τη σύσταση εμπνευσμένη από την causal inference για την αντιμετώπιση των προκαταλήψεων στα συστήματα συστάσεων [1], [3], [4].



Εικόνα 1.1.1.α: Εικονογραφημένο παράδειγμα ενός συστήματος συστάσεων
Πηγή:https://d2l.ai/chapter_recommender-systems/recsys-intro.html

1.1.2: Κατηγορίες Συστημάτων Συστάσεων

Υπάρχουν αρκετές κατηγορίες Συστημάτων Συστάσεων που χρησιμοποιούνται διάχυτα, σήμερα. Το Συνεργατικό Φιλτράρισμα (Collaborative Filtering - CF), είναι μια από τις πιο ευρέως χρησιμοποιούμενες προσεγγίσεις, το οποίο κάνει συστάσεις με βάση τη συμπεριφορά άλλων χρηστών. Το CF μπορεί να κατηγοριοποιηθεί σε προσεγγίσεις που είναι context-aware, σε προσεγγίσεις που βασίζονται στη μνήμη και ή σε μοντέλα.

Το CF που βασίζεται στη μνήμη είναι απλό και διαισθητικό, αλλά αντιμετωπίζει προκλήσεις με data sparsity και υπολογιστικές δυσκολίες σε μεγάλους χώρους χρηστών ή αντικειμένων. Ο υπολογισμός της ομοιότητας, είναι ζωτικής σημασίας στο συνεργατικό φιλτράρισμα. Οι κοινές μέθοδοι περιλαμβάνουν τον Συντελεστή Συσχέτισης Pearson

(PCC), την Ομοιότητα Συνημιτόνου (COS), τη Μέση Τετραγωνική Διαφορά (MSD), την Εγγύτητα Επίδρασης Δημοτικότητας (PIP), την Ομοιότητα Jaccard και την Εγγύτητα Σημασίας Μοναδικότητας (PSS) . Το CF που βασίζεται σε μοντέλα, χρησιμοποιεί τεχνικές μηχανικής μάθησης για να κάνει συστάσεις.

Θα αναφερθούν και άλλες προσεγγίσεις Συστημάτων Συστάσεων παρακάτω. Το Content-based RS ταξινομεί τα αντικείμενα με βάση τις πληροφορίες των χαρακτηριστικών τους και παρέχει συστάσεις ανάλογα. Αντιμετωπίζει τις συστάσεις ως ένα πρόβλημα ταξινόμησης συγκεκριμένο για τον χρήστη, μαθαίνοντας έναν ταξινομητή με βάση τα χαρακτηριστικά του αντικειμένου και παρέχει μια πιο προσωπική σύσταση. Το Content-based RS εστιάζει σε μοντέλα προτίμησης χρήστη και ιστορικό αλληλεπίδρασης χρήστη. Τα δεδομένα κειμένου και τα visual/multimedia χαρακτηριστικά είναι σημαντικές πηγές εκμάθησης χαρακτηριστικών . Επιπρόσθετα, υπαρχουν και τα Knowledge-based RS, τα οποία δεν βασίζονται στο ιστορικό αλληλεπίδρασης χρήστη-αντικειμένου και χρησιμοποιούνται σε σύνθετους τομείς όπου τα αντικείμενα δεν αγοράζονται, συνήθως. Βασική προϋπόθεση είναι η γνώση του τομέα, η οποία βοηθά στην υπέρβαση προβλημάτων, όπως της "cold start" και των "grey sheep" . Τέλος, το Hybrid RS ενσωματώνει διάφορες πηγές δεδομένων και μεθόδους για την επίτευξη συστάσεων σε πραγματικό χρόνο και ακριβείς συστάσεις. Το Deep Learning επιτρέπει την αποτελεσματική συγχώνευση πληροφοριών από πολλές πηγές, όπως το περιβάλλον (context) και το περιεχόμενο (content) .

Μια άλλη κατηγορία Συστημάτων Συστάσεων είναι τα Group Recommender Systems, τα οποία στοχεύουν στην παροχή συστάσεων σε μια ομάδα χρηστών που έχουν παρόμοιες προτιμήσεις . Σε προσεγγίσεις που βασίζονται στη μνήμη, χρησιμοποιείται συνήθως η συνάθροιση προτιμήσεων. Η συνάθροιση αποτελεσμάτων συγκεντρώνει τα αποτελέσματα των συστάσεων όλων των χρηστών. Η συνάθροιση προφίλ κάνει προσωπικές συστάσεις σε έναν εικονικό χρήστη που εξάγεται από το προφίλ των μελών της ομάδας .[1], [4], [5], [6], [7], [8], [9]

Σημαντικό είναι, να αναφερθούν κάποιες προκλήσεις, όσον αφορά τα συστήματα συστάσεων. Οι προκαταλήψεις δημοτικότητας (Popularity bias), επιλογής (selection bias), έκθεσης (exposure bias) και τοποθεσίας (location bias) πρέπει να μετριαστούν χρησιμοποιώντας τεχνικές debiasing . Η δικαιοσύνη με βάση τον χρήστη και με βάση το έργο είναι σημαντικοί παράγοντες, επίσης. Τα Adversarial attacks πρέπει να αντιμετωπιστούν για να επικυρωθεί η ανθεκτικότητα (robustness) των συστημάτων συστάσεων . Καίρια είναι, επίσης, η βελτίωση της εμπειρίας του χρήστη. Οι αξιολογήσεις που επικεντρώνονται στον χρήστη και είναι υπεύθυνες είναι ζωτικής σημασίας για την παροχή ακριβών, δίκαιων, ενημερωμένων και χρήσιμων υπηρεσιών
. Η χρήση πολυτροπικών (Multimodal) τεχνολογιών για τη μοντελοποίηση χρήστη, αντικειμένου και περιβάλλοντος είναι απαραίτητη , όπως και η έρευνα σχετικά με την αλληλεπίδραση Ανθρώπου-Υπολογιστή (HCI), για να διερευνηθεί πώς τα συστήματα μπορούν να υποστηρίξουν καλύτερα τους χρήστες στη διαδικασία λήψης αποφάσεων . Τέλος,αξίζει να αναφερθεί η συλλογή και ενσωμάτωση πληροφοριών σχετικά με τις τάσεις

που σχετίζονται με τις συνήθειες, την κατανάλωση και τις ατομικές προτιμήσεις των χρηστών .

Στον τομέα των Συστημάτων Συστάσεων, η διασφάλιση της προστασία των δεδομένων και του απορρήτου των χρηστών σε όλη τη διαδικασία σύστασης, επιτυγχάνεται μέσω διαφόρων μορφών αξιολόγησης. Τα Συστήματα Συστάσεων μπορούν να αξιολογηθούν, χρησιμοποιώντας Δείκτες Βάσει Αξιολόγησης (Rating Based Indicators - RBI), οι οποίο αξιολογούν τις συστάσεις με βάση μια προβλεπόμενη βαθμολογία αξιολόγησης και Δείκτες Βάσει Αντικειμένου (Item Based Indicators - IBI),οι οποίοι αξιολογούν τις συστάσεις με βάση ένα σύνολο ή μια λίστα προβλεπόμενων αντικειμένων . Οι κοινές μετρικές περιλαμβάνουν το Precision, το Recall και το F-Measure .[5], [10]

Καταλήγοντας, τα συστήματα συστάσεων έχουν εξελιχθεί σημαντικά, οδηγούμενα από τις τεχνολογικές εξελίξεις και μια βαθύτερη κατανόηση των αναγκών των χρηστών. Η μελλοντική έρευνα πιθανότατα θα επικεντρωθεί στην αντιμετώπιση των υπαρχουσών προκλήσεων, όπως οι προκαταλήψεις και η δικαιοσύνη, και στην ενσωμάτωση νέων τεχνολογιών για τη δημιουργία πιο εξατομικευμένων, ισχυρών και φιλικών προς τον χρήστη συστημάτων.

1.2: Ημι-εποπτευόμενη Μάθηση και Παραγοντοποίηση Πινάκων

1.2.1. Ημι-Εποπτευόμενη Μάθηση

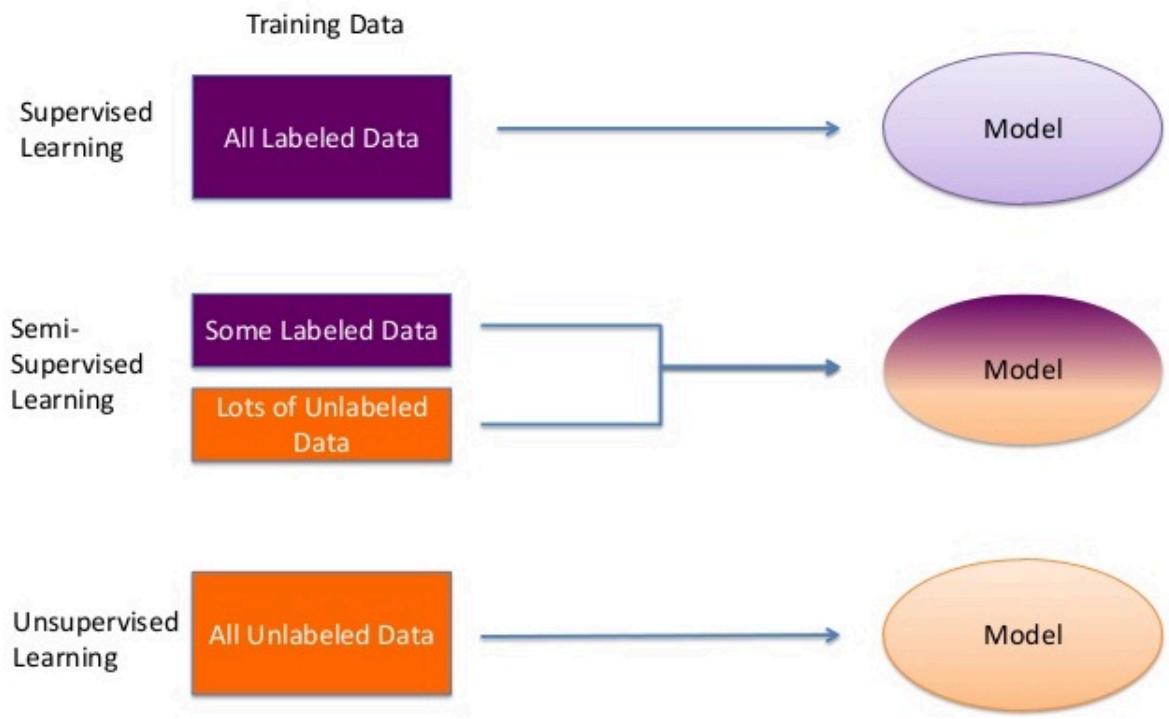
Για να γίνει εφικτή η δημιουργία προτάσεων, τα περισσότερα RS χρησιμοποιούν κάποια ή και πολλές τεχνικές Μηχανικής Μάθησης(Machine Learning-ML) .Επομένως, σε αυτό το σημείο θα ήταν χρήσιμο να γίνει μια παρουσίαση στις βασικές έννοιες που σχετίζονται με την ML και αφορούν τα RS καθώς και την παρούσα εργασία.

Με τον όρο Μηχανική Μάθηση (Machine Learning-ML), αναφερόμαστε σε ένα σύνολο αλγοριθμικών και υπολογιστικών τεχνικών υποσύνολο το ευρύ πεδίου της Τεχνητής Νοημοσύνης (AI). “Η Μηχανική Μάθηση είναι η επιστήμη που επιτρέπει στους υπολογιστές να μαθαίνουν και να δρουν όπως οι άνθρωποι, βελτιώνοντας την ικανότητά τους να μαθαίνουν με την πάροδο του χρόνου, με αυτόνομο τρόπο, μέσω της παροχής δεδομένων και πληροφοριών με τη μορφή παρατηρήσεων και αλληλεπιδράσεων με τον πραγματικό κόσμο.”[11]

Βασικό χαρακτηριστικό της ML, είναι η εστίαση στα δεδομένα ώς πρωταρχικό εργαλείο για την εκπαίδευση, αξιολόγηση και εξαγωγή συμπερασμάτων από τα μαθηματικά μοντέλα που χρησιμοποιούνται. Η ML χωρίζεται, κατα κανόνα, σε δύο βασικές κατηγορίες: την Εποπτευόμενη Μάθηση (Supervised Learning-SL) και την Μη Εποπτευόμενη Μάθηση (Unsupervised Learning-UL). Η ειδοποιός διαφορά μεταξύ των δύο αυτων κατηγορίων είναι ο τύπος των δεδομένων που χρησιμοποιούνται κατα την εκπαιδευση. Στην SL τα δεδομένα εκπαίδευσης είναι επισημασμένα. Για παράδειγμα, εάν θέλουμε να εκπαιδευσουμε ενα μοντέλο ταξινόμησης με έναν αλγοριθμο SL χρησιμοποιήσουμε δεδομένα για τα οποία γνωρίζουμε τις ετικέτες τους, δηλαδή την κατηγορία στην οποία ανήκουν. Σε ένα μοντέλο UL δεν έχουμε ετικέτες για τα δεδομένα εκπαίδευσης, επομένως

προσπαθούμε να εξάγουμε συμπεράσματα και μοτίβα απο τα δεδομενα χωρίς εναν προκαθορισμένο τρόπο.

Υπάρχει μια τρίτη κατηγορία, η Ημι-εποπτευόμενη μάθηση (Semi-Supervised Learning -SSL) ώπου για την εκπαιδευση αξιοποιείται ένα μικρό σύνολο προσημασμένων δεδομένων και ένα μεγαλύτερο σύνολο μη-προσημασμένων τα οποία χρησιμοποιούνται συνολικά για την εξαγωγή συμπερασμάτων. Αυτή η κατηγορία μας ενδιαφέρει για το αντικείμενο της πτυχιακής αυτής. Τεχνικές που εφαρμόζουν SSL, χρησιμοποιούνται συχνά στα συστήματα συστάσεων. Συνήθως, δεν υπάρχουν πληροφορίες για πολλές συσχετίσεις μεταξύ χρηστων και αντικειμένων, για παράδειγμα στα δεδομένα που θα χρησιμοποιηθούν σε αυτή την εργασία, δηλαδή κριτικές χρηστών σε αντικείμενα, οι υπαρκτές κριτικές στο σύνολο δεδομένων είναι ένα πολύ μικρό υποσύνολο των πιθανών κριτικών μεταξύ αντικειμένων και χρηστών. Αυτό, ισχύει για τα περισσότερα σύνολα δεδομένων που χρησιμοποιούνται στην ανάπτυξη αλγορίθμων στα RS καθώς και στην πρακτική εφαρμογή τους σε υπαρκτά RS με πραγματικού χρόνου δεδομένα. Συνεπώς, τεχνικές SSL είναι ιδανικές για να αξιοποιήσουν την ιδιότητα αυτή των αραιών δεδομένων.

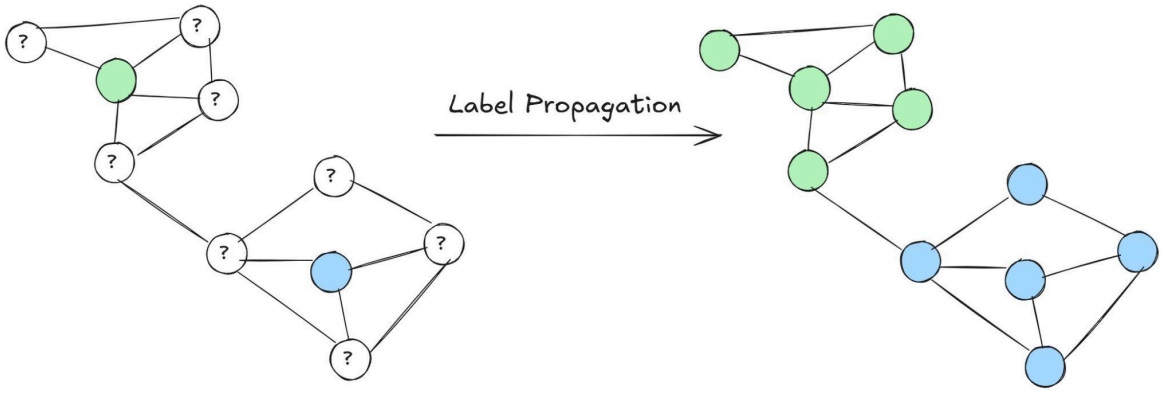


Εικόνα 1.2.1: Διαφορετικοί τύποι μηχανικής μάθησης
Πηγή:https://www.slideshare.net/slideshow/dataiku-hadoop-summit-semisupervised-learning-with-hadoop-for-understanding-user-web-behaviours/33139043

1.2.2 Label Propagation

Μια τεχνική που χρησιμοποιείται στα RS και ανήκει στην κατηγορία της SSL, είναι η τεχνική Label Propagation(LP)[12]. Η LP εφαρμόζεται σε προβλήματα που μπορούν να μοντελοποιηθούν ώς γράφοι διασυνδεδεμένων δεδομένων τα οποία θέλουμε να κατηγοριοποιήσουμε συνήθως. Ένας μικρός αριθμός των κόμβων (nodes) είναι προσημασμένα με συγκεκριμένες ετικέτες (labels),τα υπόλοιπα είναι μη προσημασμένα. Οι ακμές (edges) συμβολίζουν την ομοιότητα μεταξύ των διαφορετικών κόμβων που συνδέουν. Οι ετικέτες "διαδίδονται" από τους γνωστούς στους άγνωστους κόμβους μέσω των ακμών. Όσο πιο "κοντά" είναι δύο κόμβοι, τόσο πιο πιθανό είναι να μοιράζονται την ίδια ετικέτα. Ο αλγόριθμος λειτουργεί επαναληπτικά, σε κάθε βήμα οι κόμβοι ενημερώνουν τις ετικέτες τους με βάση τους γείτονές τους. Ο αλγόριθμος συγκλίνει και σταματάει ένα στο τελευταίο βήμα δεν υπήρξαν αλλαγές στις ετικέτες των κόμβων σε σχέση με το αμέσως προηγούμενο βήμα.[13], [14]

Η τεχνική LP έχει διάφορες εφαρμογες στον πραγματικό κόσμο οπως στην κατηγοριοποίηση κειμένου, την ιατρική καθώς και την ανάλυση κοινωνικών δικτύων. Σε σχέση τωρα με αυτή την πτυχιακή η LP μας ενδιαφέρει, καθώς αποτελεί ένα χαρακτηριστικό παράδειγμα SSL που μπορεί να χρησιμοποιηθεί στο Συνεργατικό Φιλτράρισμα βασισμένο στους χρήστες, ορίζοντας κάποια μετρική (cosine similarity, pearson correlation) για να υπολογίσουμε την ομοιότητα μεταξύ χρηστών (προσημασμενων και μη), δηλαδή τις ακμές ενός γραφήματος με κόμβους τους ίδιους τους χρήστες. Από τις ακμές μεταξύ χρηστών, μπορούμε να κατασκευάσουμε μία μήτρα γειτνίασης που θα περιγράφει την ομοιότητα μεταξύ χρηστών. Αυτές οι μήτρες είναι ισοδύναμες με τους πίνακες εγγύτητας που θα αναλύσουμε την συνέχεια.



Εικόνα 1.2.2: Label Propagation
Πηγή:https://www.nb-data.com/p/label-propagation-in-hybrid-models

1.2.3 Παραγοντοποίηση Πινάκων

Ένα σύνολο μεθόδων που χρησιμοποιούνται συχνά στα RS και δεν αναφέρθηκαν παραπάνω είναι οι τεχνικές Παραγοντοποίησης Πινάκων (Matrix Factorization - MF). Μοιάζουν με το συνεργατικό φιλτράρισμα στο γεγονός ότι αξιοποιούν παρελθοντικά

δεδομένα από τις προτιμήσεις των χρηστών αλλά επικεντρώνονται στην μείωση του μεγέθους του πίνακα εγγύτητας διατηρώντας όσο το δυνατόν περισσότερη πληροφορία.Συχνά εφαρμόζουν SSL καθώς εκμεταλλεύονται τον μεγάλο όγκο μη προσημασμένων δεδομένων μεταξύ χρηστων-αντικειμένων.Μια τέτοια μέθοδος παραγοντοποίησης πινάκων θα αναπτυχθεί στην συνέχεια στον αλγόριθμο GenRec0 . [6], [14], [15]

1.3: Γενετικοί Αλγόριθμοι

Οι Γενετικοί Αλγόριθμοι (Genetic Algorithms και εν συντομία, GA) αποτελούν έναν τύπο αλγορίθμων αναζήτησης και βελτιστοποίησης, που έχουν επηρεαστεί από την θεωρία της φυσικής επιλογής καθώς και την επιστήμη της γενετικής. Πρωτοεμφανίστηκαν την δεκαετία του 1970 απο τον Jhon Holland[16]. Προσομοιώνουν την διαδικασία της εξέλιξης ενός είδους με σκοπό την εύρεση βέλτιστων λύσεων σε διάφορα υπολογιστικά προβλήματα. Ανήκουν στην γενικότερη κατηγορία των εξελικτικών αλγορίθμων και μεθόδων που συχνά επιλέγονται για πολύπλοκα προβλήματα βελτιστοποίησης και αναζήτησης.

Οι ΓΑ (Γενετικοί Αλγόριθμοι) μιμούνται φυσικές διαδικασίες, αξιοποιώντας ένα σύνολο πιθανών λύσεων, το οποίο ορίζεται ώς πληθυσμός. Στόχος είναι, η κατάληξη σε βέλτιστη λύση με βάση το πρόβλημα που εξετάζεται κάθε φορά. Μέσω μιας επαναληπτικής διαδικασίας, προσομοιώνεται η αναπαραγωγή καθώς και η “φυσική επιλογή” στον πληθυσμό με το πέρασμα των γενίων και του χρόνου. Ο πληθυσμός (Population) στους ΓΑ, αποτελείται από ένα διακριτό σύνολο ατόμων (Individuals), όπου το κάθε άτομο συμβολίζει μία πιθανή λύση στο πρόβλημα. Σκοπός είναι, με κάθε νέα γενιά (Generation), δηλαδή με κάθε επανάληψη, τα άτομα του πληθυσμού να αποτελούν βελτιωμένες λύσεις σε σχέση με κάθε προηγούμενη γενιά, μέχρις ότου οι λύσεις να συγκλίνουν. Ο τρόπος που υπολογίζεται η ποιότητα των λύσεων, αξιολογώντας τα άτομα, είναι μέσω μιας Συνάρτηση Καταλληλότητας (Fitness Function). Οι συναρτήσεις καταλληλότητας μπορεί να διαφέρουν αρκετά μεταξύ διαφορετικών υλοποιήσεων γενετικών αλγορίθμων, και εξαρτώνται πάντα από το πρόβλημα που λύνει ο κάθε ΓΑ. Με αυτό τον τρόπο, ο πληθυσμός αξιολογείται και για κάθε άτομο υπολογίζεται μια τιμή καταλληλότητας (Fitness value). Τεχνικά, ο όρος άτομο περιλαμβάνει δύο στοιχεία, το χρωμόσωμα (Chromosome), οπου συμβολίζει μια πιθανή λύση στο πρόβλημα και την τιμή καταλληλότητας, που υπολογίζεται με την συνάρτηση καταλληλότητας. Βέβαια, ο όρος άτομο θα χρησιμοποιηθεί στην συνέχεια για να περιγράψει και τις δύο έννοιες.

Ένας ΓΑ αξιολογεί διαφορετικά άτομα σε κάθε γενιά, τα άτομα αυτά προκύπτουν εφαρμόζοντας διάφορες τεχνικές στον πληθυσμό της προηγούμενης γενιάς. Οι τεχνικές αυτές ονομάζονται Γενετικοί Τελεστές και έχουν επηρεαστεί από όμοιες διαδικασίες που συμβαίνουν κατα την βιολογική εξέλιξη. Οι Συντελεστές αυτοί είναι οι εξής:

Γενετικοί Τελεστές	Κυριότερες Μέθοδοι
Τελεστής Επιλογής (Selection Operator): καθορίζει ποια άτομα (λύσεις) θα αναπαραχθούν στην επόμενη γενιά.	1)**Ρουλέτα** (Roulette Wheel Selection): Πιθανότητα επιλογής ανάλογη με το fitness. 2)**Τουρνουά** (Tournament Selection): Επιλογή του καλύτερου από τυχαία υποομάδα. 3)**Ελιτισμός** (Elitism Selection): Οι καλύτερες λύσεις περνούν αυτόματα στην επόμενη γενιά. 4)**Κατάταξη** (Rank Selection): Τα άτομα κατατάσσονται και επιλέγονται βάσει θέσης. 5)**Σταθερή Αναλογία** (Steady-State Selection): Αντικατάσταση μόνο λίγων ατόμων ανά γενιά.
Τελεστής Διασταύρωσης (Crossover Operator): Συνδυάζει δύο γονείς για τη δημιουργία απογόνων.	1)**One-point crossover:** Κόβει το γονιδίωμα σε ένα σημείο και ανταλλάσσει κομμάτια. 2)**Two-point crossover**: Κόβει σε δύο σημεία και ανταλλάσσει το ενδιάμεσο τμήμα. 3)**Uniform crossover**:Τυχαία επιλογή γονιδίων από κάθε γονέα. 4)**Arithmetic crossove**r: απόγονοι είναι γραμμικός συνδυασμός των γονέων.
Τελεστής Μετάλλαξης (Mutation Operator): Τροποποιεί τυχαία γονίδια των απογόνων για διατήρηση της ποικιλομορφίας.	1)**Bit-flip mutation:** Αλλάζει τυχαία ένα bit (για δυαδική αναπαράσταση). 2)**Swap mutation**: Ανταλλάσσει δύο γονίδια. 3)**Gaussian mutation**: Προσθέτει μικρή τυχαία τιμή σε συνεχείς μεταβλητές. 4)**Inversion mutation**: Αντιστροφή σειράς γονιδίων.
Τελεστής Αντικατάστασης (Replacement Operator): Καθορίζει ποια άτομα θα παραμείνουν στην επόμενη γενιά.	1)**Generational replacement** : Όλος ο πληθυσμός αντικαθίσταται κάθε γενιά. 2)**Steady-state replacement**: Μόνο μερικά άτομα αντικαθίστανται κάθε φορά. 3)**Elitism**: Οι καλύτερες λύσεις δεν διαγράφονται ποτέ.

Πίνακας 1.3.α: Γενετικοί τελεστές και μέθοδοι

Οι τελεστές αυτοί μπορούν να διαφέρουν αρκετά μεταξύ ΓΑ και εξαρτώνται άμεσα από το προβλήματα προς επίλυση, καθως και την μορφή των χρωμοσωμάτων που εξελίσσονται. Μέσω της χρήσης των τελεστών αυτών οι ΓΑ έχουν την δυνατότητα να αναζητούν και να βρίσκουν βέλτιστες λύσεις ακόμα και σε προβληματα με ιδιαίτερα πολύπλοκο χώρο αναζήτησης. Είναι χρήσιμοι για μια ευρεία γκάμα προβλημάτων βελτιστοποίησης και μπορούν να εκτελεστούν αξιοποιώντας τεχνικές παράλληλου υπολογισμού αυξάνοντας την απόδοση.

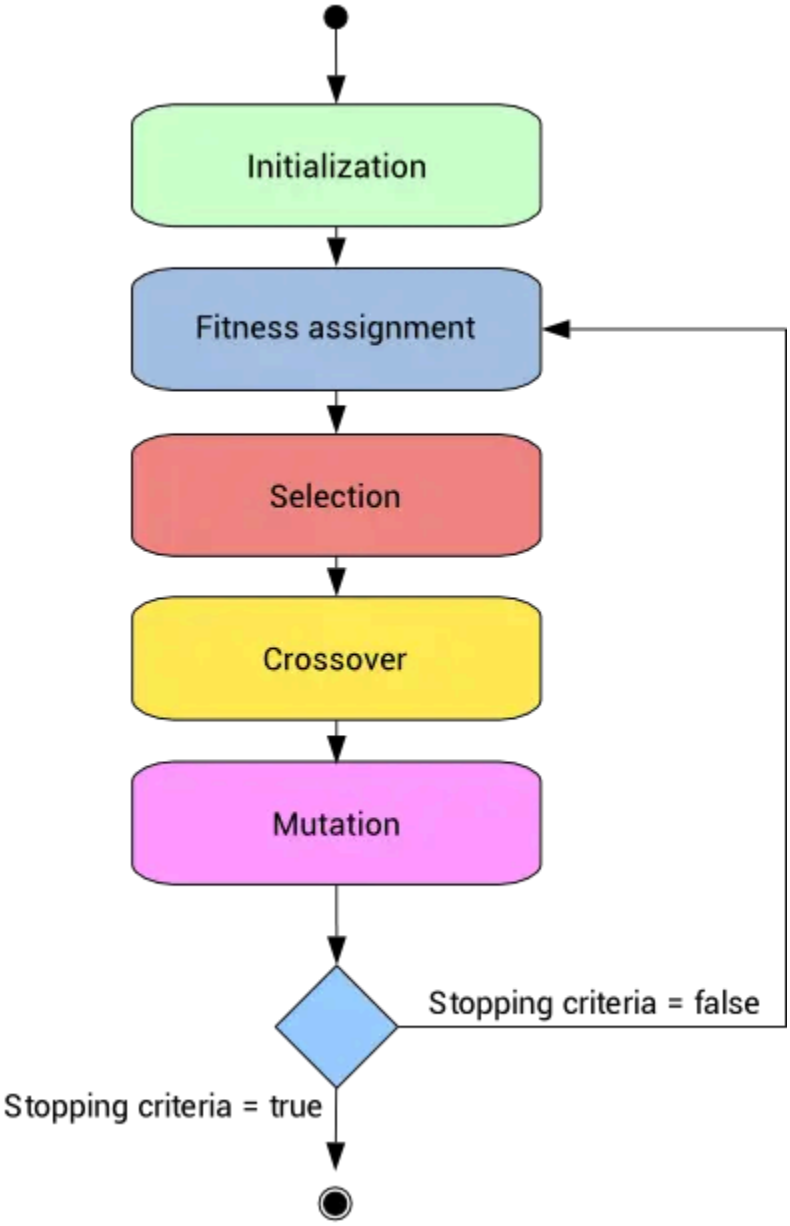
Βέβαια, ο σχεδιασμός ενός ικανού γενετικού αλγορίθμου απαιτεί την εύρεση μιας συνάρτησης καταλληλότητας που ταιριάζει στο κάθε πρόβλημα, ένα δύσκολο εγχείρημα για πολλές εφαρμογες. Επίσης, οι διάφορες παράμετροι, όπως ο βαθμός μετάλλαξης (mutation rate), το μέγεθος του πληθυσμού και ο αριθμός των γενεών επηρεάζουν σημαντικά τα τελικά αποτελέσματα [17].

Οι γενετικοί αλγόριθμοι χρησιμοποιούνται για την επίλυση πολύπλοκων προβλημάτων σε διάφορα ερευνητικά πεδία. Στον παρακάτω πίνακα παρουσιάζονται κάποιες από αυτές τις εφαρμογές ΓΑ.

Optimization Problems (Προβλήματα Βελτιστοποίησης)	Οι Γενετικοί Αλγόριθμοι (ΓΑ) χρησιμοποιούνται για τη μεγιστοποίηση ή την ελαχιστοποίηση συναρτήσεων αντικειμενικού σκοπού σε διάφορους τομείς, όπως η μηχανική, τα οικονομικά και η επιχειρησιακή έρευνα.
Machine Learning (Μηχανική Μάθηση)	Οι ΓΑ εφαρμόζονται στον σχεδιασμό και τη βελτιστοποίηση νευρωνικών δικτύων, στη βελτίωση αλγορίθμων ταξινόμησης και στην ανάπτυξη μοντέλων μηχανικής μάθησης.
Image Processing (Επεξεργασία Εικόνας)	Οι ΓΑ χρησιμοποιούνται για την τμηματοποίηση εικόνας, την εξαγωγή χαρακτηριστικών και την αναγνώριση εικόνων.
Robotics (Ρομποτική)	Οι ΓΑ εφαρμόζονται στον προγραμματισμό τροχιάς ρομπότ, στα συστήματα ελέγχου και στον σχεδιασμό ρομπότ.
Scheduling (Χρονοπρογραμματισμός)	Οι ΓΑ χρησιμοποιούνται για την επίλυση προβλημάτων χρονοπρογραμματισμού

	στη βιομηχανία, τις μεταφορές και την κατανομή πόρων.
Internet of Things - IoT (Διαδίκτυο των Πραγμάτων)	Οι ΓΑ εφαρμόζονται στην επιλογή έξυπνων συσκευών, στη μετάδοση δεδομένων και στην εξόρυξη πληροφοριών σε δίκτυα IoT.
Traffic Management (Διαχείριση Κυκλοφορίας)	Οι ΓΑ χρησιμοποιούνται σε έξυπνα συστήματα σηματοδότησης για τη μείωση του χρόνου ταξιδιού και των εκπομπών ρύπων.
Cloud Computing (Υπολογιστικό Νέφος)	Οι ΓΑ εφαρμόζονται στην εξισορρόπηση φορτίου και στον χρονοπρογραμματισμό εργασιών σε περιβάλλοντα cloud computing.
Bioinformatics (Βιοπληροφορική)	Οι ΓΑ χρησιμοποιούνται για την ανάλυση δομών DNA και την επίλυση πολύπλοκων προβλημάτων στη βιοπληροφορική.

Πίνακας 1.3.β: Εφαρμογές Γενετικών Αλγορίθμων



Εικόνα 1.3.α: Διαδικασία Γενετικού αλγορίθμου
Πηγή:https://www.neuraldesigner.com/blog/genetic_algorithms_for_feature_selection/

Κεφάλαιο 2: Μεθοδολογία

2.1: Ορισμός Προβλήματος και Βασική ιδεα

Ξεκινώντας, θα πρέπει να οριστεί το προβλημα, για το οποίο οι μέθοδοι που θα αναπτυχθούν στην συνεχεια καλούνται να βρουν ικανοποιητικές λύσεις. Το πρόβλημα που θα λυθεί, είναι στον πυρήνα του παλινδρόμηση στοχεύοντας στην πρόβλεψη των αξιολογήσεων χρηστών σε αντικείμενα. Αξιοποιώντας πληροφορίες από κριτικές όμοιων χρηστών, εφαρμόζοντας δηλαδή **Συνεργατικό Φιλτράρισμα Βασισμένο στους Χρήστες**. Στόχος είναι, η εύρεση μιας βέλτιστης σύνδεσης χρηστών χρησιμοποιώντας μεθόδους Τεχνητής Νοημοσύνης και συγκεκριμένα, Γενετικούς Αλγόριθμους.

Έστω U **= {**u_1**,** u_2**, … ,** u_n**}** το σύνολο των χρηστών , με |U| **=** n, και ***I*** **= {**i_1**,** i_2 **, …,** i_n**}** το σύνολο των στοιχείων, με |I| = m. Μπορούμε να ορίσουμε μια μήτρα αξιολογήσεων $R \in R^{n \times m}$, όπου το στοιχείο R_{ui} αντιπροσωπεύει την αξιολόγηση του χρήστη u για το στοιχείο.

Η μήτρα R είναι μια αραιή μήτρα, όπως θα φανεί και με την ανάλυση του συνόλου δεδομένων, καθως και όπως είναι το σύνηθες με τέτοια σύνολα δεδομένων. Οι περισσότεροι χρήστες u δεν θα έχουν αξιολογήσει την πλειονότητα των αντικειμένων i **.** Το σύνολο των αξιολογήσεων που γνωρίζουμε a, είναι κατα πολύ μικρότερο του συνολικού αριθμού των πιθανών αξιολογήσεων (a << $n \times m$). Όπως με κάθε μέθοδο συνεργατικού φιλτραρίσματος αξιοποιούμε την παραδοχή ότι οι χρήστες που έχουν κοινά ενδιαφέροντα ,δηλαδή έχουν επαφή και έχουν αξιολογήσει παρόμοια αντικείμενα, με αντίστοιχο τρόπο ,άρα, η κριτική τους για ένα αντικείμενο i θα είναι η ίδια ή δεν θα διαφέρει σημαντικά, θα αξιολογήσουν παρόμοια και αντικείμενα για τα οποία δεν έχουμε κριτικές ακόμα.

Επομένως, θα προσπαθήσουμε να ενώσουμε όμοιους χρήστες με έναν βέλτιστο τρόπο. Ορίζουμε μια μήτρα κοινωνικών διαδράσεων ή μήτρα εγγύτητας (affinity matrix) μεταξύ χρηστών, **W** $\in R^{n \times m}$, με το στοιχείο w_{uv} να εκφράζει την κοινωνική σχέση μεταξύ των χρηστών. Όσο μεγαλύτερη είναι η τιμή του w_{uv} , τόσο μεγαλύτερη η συσχέτιση μεταξύ των u και υ. Σκοπός λοιπόν, είναι να καταλήξουμε σε μια τέτοια μορφή για τον W, έτσι ώστε να περιγράφει την σχέση μεταξύ χρηστών με βέλτιστο τρόπο , και αξιοποιόντας την πληροφορία που περιέχεται στο W να μπορούμε να προβλέψουμε το πώς ο κάθε χρήστης θα βαθμολογήσει το κάθε αντικείμενο.

Στην συνέχεια θα αναπτύξω δύο μοντέλα που θα βελτιστοποιούν την μορφή αυτής της μητρας εγγύτητας W, χρησιμοποιώντας γενετικούς αλγορίθμους. Το κάθε ένα από τα δύο, θα εφαρμόζει διαφορετικές τεχνικές για να εξελίξει μια βέλτιστη μορφή του πίνακα W.

2.2: Παρόμοιες προσπάθειες πάνω στο αντικείμενο

Με σκοπό την ανάδειξη βέλτιστων μεθόδων για το παραπάνω πρόβλημα έχουν γίνει πολλαπλές προσπάθειες στην ακαδημαϊκή βιβλιογραφία.

Καταρχάς υπάρχουν οι καθιερωμένες μέθοδοι βασισμένες στην Παραγοντοποίηση Πινάκων Εγγύτητας(MF) που αξιοποιούνται συχνά στα RS κάνοντας χρήση διαφόρων τεχνικών ΜΜ. Για παράδειγμα η τεχνική SVD(Singular Value Decomposition)[12] χρησιμοποιεί μεθόδους γραμμικής άλγεβρας για να πετύχει την παραγοντοποίηση, η μέθοδος FM(Factorization Machine)[15], χρησιμοποιεί Support Vector Machines(SVM) ενω το μοντέλο DeepFM χρησιμοποιεί νευρωνικά δίκτυα(Neural Networks) και βαθιά μάθηση(Deep Learning)[18].[15]

Τέλος, σύμφωνα με την βιβλιογραφία βέβαια υπάρχουν και άλλες μέθοδοι που κάνουν χρήση Γενετικων Αλγορίθμων στα Συστήματα συστάσεων και συγκεκριμένα στην πρόβλεψη αξιολογησεων αξιοποιώντας την ευελιξία και την ικανότητα των GA .[19], [20], [21]

2.3: Μαθηματικό Υπόβαθρο

Τα δύο μοντέλα που θα αναπτυχθούν έχουν κοινο το γεγονός ότι χρησιμοποιούν γενετικό αλγόριθμο για την βελτιστοποίηση των λύσεων στο πρόβλημα που αναζητουν, διαφέρουν όμως στον τρόπο που διαχειρίζονται τον W.

2.3.1 Ημι Εποπτευόμενη Μάθηση σε πίνακες εγγύτητας

Η προσέγγιση που ακολούθησα στη δημιουργία αυτού του μοντέλου εστιάζει στην μορφή του πίνακα εγγύτητας W, αξιοποιώντας υποπίνακες σχετικούς με κάθε αντικείμενο.

Για κάθε στοιχείο $i \in \boldsymbol{I}$**:**

• **Χρήστες που έχουν αξιολογήσει το** i , γνωστές δηλαδή αξιολογήσεις:

$m_i \subseteq U$**,** με $\left|m_i\right|$ **=** n_1

• **Χρήστες που δεν έχουν αξιολογήσει το** i , άγνωστες δηλαδή αξιολογήσεις:

$m_{-i} \subseteq U$**,** με $\left|m_{-i}\right|$ **=** n_{-1}

Έχοντας αυτά τα στοιχεία, μπορούμε να κατασκευάσουμε για κάθε αντικείμενο υπομήτρες που θα περιέχουν σχέσεις μεταξύ χρηστών που το έχουν αξιολογήσει, τους οποίους ονομάζουμε γνωστούς χρήστες και αυτών που δεν το έχουν αξιολογήσει, τους οποίους ονομάζουμε γνωστούς χρήστες.

• Η **Υπομήτρα** $W_u \in R^{n_{-i} \times n_{-i}}$ (αγνώστων προς αγνώστους) περιέχει τις διαδράσεις μεταξύ χρηστών που **δεν** έχουν αξιολογήσει το i :

$$W_u = W[\, m_{-i}, m_{-i}\,]$$

• Η **Υπομήτρα** $W_i \in R^{n_{-i} \times n}$ (γνώστων προς αγνώστους) που περιέχει τις διαδράσεις από χρήστες που **έχουν** αξιολογήσει το i προς χρήστες που δεν το έχουν αξιολογήσει:

$$W_l = W[\, m_{-i}, m_i\,]$$

Ο στόχος είναι να προβλέψουμε τις άγνωστες αξιολογήσεις για τους χρήστες m_{-i}, βασιζόμενοι:

• Στις γνωστές αξιολογήσεις .
• Στις κοινωνικές διαδράσεις W_u και W_l .

Οι προβλεπόμενες αξιολογήσεις $R_{i,pred}$ καθορίζονται από την σχέση:

$$R_{i,pred} = W_u R_{i,pred} + W_l R_{mi}$$

• $W_u R_{i,pred}$: Επίδραση μεταξύ αγνώστων χρηστών.
• $W_l R_{m1}$: Επίδραση από γνωστούς προς αγνώστους χρήστες.

Καθώς σκοπεύουμε στην επίλυση για $R_{i,pred}$ **:**

Αφαιρούμε τον όρο $W_u R_{i,pred}$ **:**

$$R_{i,pred} - W_u R_{i,pred} = W_l R_{mi}$$

Βγάζουμε κοινό παράγοντα:

$$(\boldsymbol{I} - W_u) R_{i,pred} = W_l R_{mi}$$

Προσθήκη Κανονικοποίησης (Ridge):

Για ευστάθεια και αποφυγή υπερεκπαίδευσης, προσθέτουμε τον όρο $\lambda \boldsymbol{I}$**.**

$$(\boldsymbol{I} - W_u + \lambda \boldsymbol{I}) R_{i,pred} = W_l R_{mi}$$

Καταλήγουμε στον τελικό τύπο:

$$R_{i,pred} = (\boldsymbol{I} - W_u + \lambda \boldsymbol{I})^{-1} W_l R_{mi} \quad (1)$$

• $(\boldsymbol{I} - W_u + \lambda \boldsymbol{I})^{-1}$: Διορθώνει την επίδραση των αγνώστων χρηστών και εισάγει σταθερότητα.

• $W_l R_{mi}$: Μεταφέρει τις πληροφορίες από γνωστές αξιολογήσεις σε άγνωστες.

Οι προβλέψεις για το $R_{i,pred}$ εξαρτώνται από τις υπάρχοντες κριτικές και απο την μορφή του πίνακα W. Επομένως, μπορούμε να χρησιμοποιήσουμε αυτόν τον τύπο ώς μια μέθοδο αξιολόγησης για διαφορετικους W .[14]

2.3.2: Low Rank Embeddings

Απο την προηγούμενη ανάλυση, είχαμε τον τύπο:

$$(\boldsymbol{I} - W_u + \lambda \boldsymbol{I}) R_{i,pred} = W_l R_{mi}$$

Όπου:

• $W_u = W[m_{-i}, m_{-i}]$: Διαδράσεις μεταξύ αγνώστων χρηστών.

• $W_l = W[m_{-i}, m_i]$: Διαδράσεις από γνωστούς προς αγνώστους.

Ορίζουμε το Low-Rank User Embeddings (U):

Κάθε άτομο στον πληθυσμό είναι μια μήτρα $U \in R^{n_{users} \times RANK}$, όπου RANK η διάσταση των **embeddings** (υπερπαράμετρος). Οι στήλες του U αντιπροσωπεύουν **latent factors** των χρηστών, που μαθαίνονται μέσω της γενετικής διαδικασίας.

Υπολογισμός Affinity Matrix (W) ως Γινόμενο Embeddings:

Για ένα στοιχείο i, υπολογίζουμε τον πίνακα συγγένειας μεταξύ **γνωστών** m_i και **αγνώστων** m_{-i} χρηστών ως εξής:

$$W_{subnet} = U[m_{-i}]U[m_i]^T$$

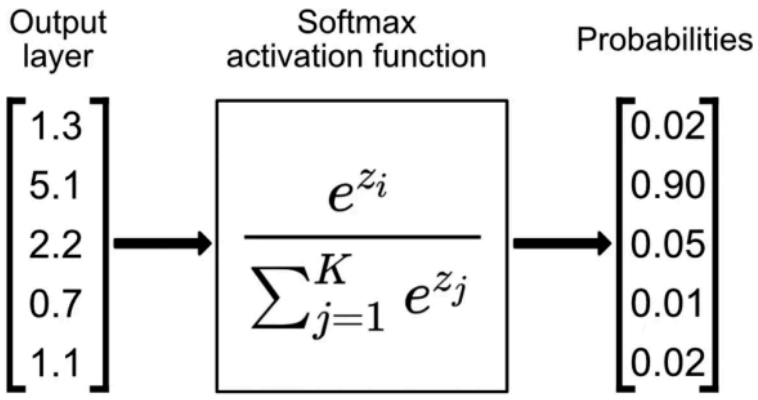
Όπου:

- $U[m_{-i}] \in R^{n_{-i} \times RANK}$
- $U[m_i] \in R^{n_i \times RANK}$

Καταλείγουμε, στον παρακάτω τύπο πρόβλεψης αξιολογήσεων:

$$R_{i,pred} = \text{softmax}\left(U_{m-i} \quad U_{mi}^T\right) R_{mi} \quad (2)$$

Όπου ο πίνακας W_{subnet} προσεγγίζεται μέσω του γινομένου **embeddings**.



Εικόνα 2.3.2.α:Συνάρτηση SOFTMAX

Η μέθοδος αυτή αποτελεί μία μέθοδο παραγοντοποίησης πινάκων (Matrix Factorization) [6], [15], οι μέθοδοι αυτοί παρουσιάστηκαν στο κεφάλαιο 1.1 .

2.4: Περιγραφή Διαδικασίας

Η αναλυτική διαδικασία με την οποία οι δύο γενετικοι αλγοριθμοι βελτιστοποιούν την μορφή της μήτρας εγγύτητας W, περιγράφεται στις παρακάτω υποενότητες. Αποτελούν

δύο πρωτότυπες μεθόδους εφαρμόζοντας συνεργατικό φιλτράρισμα βασισμένο στους χρήστες.

2.4.1: Αναλυτική διαδικασία GenRec1

Χρησιμοποιώντας τον τύπο που αναλύσαμε παραπάνω:

$$R_{i,pred} = (\boldsymbol{I} - W_u + \lambda \boldsymbol{I})^{-1} W_l R_{mi} \quad (1)$$

Δημιουργούμε έναν μηχανισμό αξιολόγησης πινάκων τύπου W . Για κάθε αντικείμενο μπορούμε να χωρίσουμε τους χρήστες που το έχουν αξιολογήσει σε δύο σύνολα, αυτούς που θα θεωρηθουν άγνωστοι κατα τον υπολογισμό του $R_{i,pred}$ και αυτούς που παραμένουν γνωστόι. Με αυτόν τον τρόπο, μπορούμε να συγκρίνουμε τα αποτελέσματα με τις πραγματικές κριτικές και να υπολογίσουμε το σφάλμα. Ως μετρική σφάλματος, επιλέχθηκε το Μέσο Τετραγωνικό Σφάλμα (MEAN SQUARED ERROR), καθώς οι μεγαλύτερες αποκλίσεις από την πραγματική τιμή “τιμωρούνται” αυξάνοντας το σφάλμα τετραγωνικά. Αυτή η τεχνική ανήκει στην κατηγορία της ημι-εποπτευόμενης μάθησης, καθώς για την πρόβλεψη της πραγματικής τιμής, δηλαδή την εκτέλεση παλινδρόμησης, χρησιμοποιούνται δεδομένα από γνωστούς και άγνωστους χρήστες.

Εφαρμόζοντας την διαδικασία αυτή για κάθε αντικείμενο του συνόλου δεδομένων και υπολογίζοντας τον μέσο όρο των σφαλμάτων, καταλήγουμε σε ένα μέτρο αξιολόγησης, με το οποίο μπορεί να πραγματοποιηθεί σύγκριση μεταξύ διαφορετικων affinity matrices. Επιλέγοντας αυτή την μέθοδο ώς συνάρτηση καταλληλότητας, μπορούμε να αναπτύξουμε έναν γενετικό αλγόριθμο που να εξελίσσει βελτιστες μορφές του πίνακα W αλλάζοντας σταδιακά την τιμή των στοιχείων του. Αυτή είναι η βασική ιδέα πίσω από την ανάπτυξη του μοντέλου GenRec1.

Για την εκκίνηση του GA ορίζεται ένα πληθυσμός P (population) που αποτελείται απο ατομα I (individuals). Το κάθε άτομο είναι μια μήτρα W μεγέθους $n \times n$ (όπου n είναι ο αριθμός των χρηστών). Οι τιμές μέσα στο W κυμαινονται απο 0.0 έως 1.0, και ο πίνακας είναι πάντα συμμετρικός. Οι τιμές του πίνακα ερμηνεύονται ώς γονίδια (genes) τα οποία αλλάζουν κατά την επαναληπτική διαδικασία της εξέλιξης. Η δημιουργία του πληθυσμού μπορεί να πραγματοποιηθεί είτε αρχικοποιώντας τυχαίες μήτρες, είτε χρησιμοποιώντας κάποια στοχαστική μέθοδο προκειμένου να αντιμετωπιστεί το πρόβλημα του cold start και ο αλγόριθμος να ξεκινήσει ήδη με κάποια βελτιωμένα γονίδια. Η ύπαρξη ενός πληθυσμού είναι προϋπόθεση για την περάτωση της κάθε γενιάς (generation).[22]

Στη συνέχεια, τα άτομα του πληθυσμού αξιολογούνται με την διαδικασία που περιγράφει παραπάνω και ταξινομούνται σε σχέση με αυτό. Υπολογίζεται και ένας δείκτης ομοιότητας μεταξύ ατόμων που στοχεύει στην “τιμωρία” των πολύ όμοιων λύσεων. Συμπεριλαμβάνοντας και τον δείκτη ομοιότητας, υπολογίζεται η τελική κατάταξη των ατόμων αυτής της γενιάς. Επιλέγονται τυχαία για το επόμενο βήμα του ζευγαρώματος από

τους κορυφαίους n πίνακες (tournament selection). Το βήμα της επιλογής (selection) ολοκληρώθηκε.

Σειρά τώρα έχουν οι διάφοροι γενετικοί τελεστές.Η μέθοδος που επιλέχθηκε ως γενετικός τελεστής διασταύρωσης για τον GenRec1 εφαρμόζει τυχαία επιλογή γονιδίων, δηλαδή θέσεων στους δύο πίνακες γονείς W, διατηρώντας την συμμετρια στους απογόνους τους. Εφαρμόζεται μετάλλαξη (mutation) στους απογόνους για την δημιουργία νέων γονιδίων που δεν έχει εξερευνήσει ο ακομα ο αλγόριθμος. Επιλέγονται λιγοι απο τους πίνακες στον παλιό πληθυσμό με το ελάχιστο σφάλμα (elitism). Σε αυτή την φάση, έχει δημιουργηθεί ο πληθυσμός για την επόμενη γενιά.

Η διαδικασία αυτή επαναλαμβάνεται για τις επόμενες γενιές κάθε φορά με διαφορετικό πληθυσμό, μειώνοντας το σφάλμα και εξελίσσοντας καλύτερες λύσεις για τον πίνακα W.

2.4.2: Αναλυτική διαδικασία GenRec0

Όμοια με τον GenRec1, ο GenRec0 είναι ένας γενετικός αλγόριθμος που εξελίσει ιδανικούς πίνακες εγγύτητας μεταξύ χρηστών. Η ειδοποιός διαφορά όμως, βρίσκεται στον τρόπο διαχείρισης των μητρών αυτών. Ο GenRec0 εξελίσει στην πραγματικότητα χαμηλού βαθμού πίνακες που περιέχουν υπερπαραμέτρους (embeddings) για κάθε χρήστη, τόσους όσους ορίζει η παράμετρος RANK. Με αντίστοιχο τρόπο, όπως υπολογίστηκαν οι προβλέψεις τον αξιολογήσεων των χρηστών στο κάθε αντικείμενο στον GenRec1, υπολογίζονται και οι προβλεψεις με την χρήση της συνάρτησης (2).

$$R_{i,pred} = \text{softmax}\left(U_{m-i} \quad U_{mi}^{T}\right) R_{mi} \quad (2)$$

Η σημαντική διαφορά και αυτό που προσδίδει ιδιαίτερη βελτίωση στην χρονική πολυπλοκότητα, είναι η εμφανώς μικρότερη διάσταση των μήτρων που χρησιμοποιούνται, καθώς και ο λιγότερο σύνθετος πολλαπλασιασμός πινάκων, που δεν χρειάζεται δηλαδή αντιστροφή.

Αφού έχουν υπολογιστεί οι εκτιμησεις, συγκρίνονται με τις πραγματικες τιμές και υπολογίζεται το MSE για κάθε αντικείμενο. Ο μέσος όρος των σφαλμάτων αυτών, είναι ο δείκτης καταλληλότητας για το άτομο (πίνακας U) που εξετάζεται και συνάρτηση καταλληλότητας η διαδικασία που περιγράφει.

Στην συνέχεια ο αλγόριθμος συμπεριφέρεται αντίστοιχα με τον Genrec1 χρησιμοποιώντας βέβαια διαφορετικούς γενετικους τελεστές, κατάλληλους για την διαφορετική μορφή των χρωμοσωμάτων. Για την διασταύρωση (crossover χρησιμοποιείται) uniform crossover με την χρήση μασκας (mask). Προκύπτουν απογόνοι και σε αυτούς εκτελείται μετάλλαξη, για την δημιουργία του νέου πληθυσμού, εφαρμόζοντας elitism κρατώντας τους καλύτερους individuals.

Η διαδικασία αυτή επαναλαμβάνεται για τις επόμενες γενιές κάθε φορά με διαφορετικό πληθυσμό, μειώνοντας το σφάλμα και εξελίσσοντας καλύτερες υπερπαραμέτρους (embeddings) του πίνακα W.

GenRec1	GenRec0
Χρησιμοποιεί γενετικό αλγόριθμο για βελτιστοποίηση σε πίνακες εγγύτητας	Χρησιμοποιεί γενετικό αλγόριθμο για βελτιστοποίηση Low Rank Embeddings(LRU) πινάκων εγγύτητας
Υπολογίζει προβλεψεις για τις κριτικές χρηστών σε αντικείμενα με την συναρτηση: $R_{i,pred} = (\boldsymbol{I} - W_u + \lambda \boldsymbol{I})^{-1} W_l R_{mi}$ (1)	Υπολογίζει προβλεψεις για τις κριτικές χρηστών σε αντικείμενα με την συναρτηση: $R_{i,pred} = \text{softmax}\left(U_{m-i} \; U_{mi}^{T}\right) R_{mi}$ (2)
Για κάθε αντικείμενο γίνεται επίλυση ενός πολύπλοκου γραμμικού συστήματος	Για κάθε αντικείμενος υπολογίζεται ένα αρκετά πιο απλό γινόμενο πινάκων
Χρειάζεται αντιστροφή μητρών	Δεν χρειάζεται αντιστροφή μητρών
Για κάθε πρόβλεψη εμπεριέχονται όλες οι συσχετίσεις σχετικών χρηστών	Για κάθε πρόβλεψη εμπεριέχονται οι υπερπαραμέτροι του κάθε σχετικού χρήστη
Αλγοριθμικά αργή και πολύπλοκη διαδικασία. Χρονική Πολυπλοκότητα: $O\left(P \cdot n^3\right)$, με P: μέγεθος πληθυσμού και n: αριθμός χρηστών	Αρκετά απλουστευμένη και ταχύτερη διαδικασία. Χρονική Πολυπλοκότητα:$O\left(P \cdot n^2\right)$, με P: μέγεθος πληθυσμού και n: αριθμός χρηστών

Πίνακας 2.4.2: Σύγκριση GenRec1 και GenRec0

2.5: Τεχνολογίες που Χρησιμοποιήθηκαν

Για τους σκοπούς της παρούσας προπτυχιακής εργασίας χρησιμοποιήθηκαν διάφορες τεχνολογίες για την ανάλυση, τον καθαρισμό και την αναπαράσταση τον δεδομένων και αποτελεσμάτων που προέκυψαν καθώς και για την εφαρμογή μεθόδων τεχνητής νοημοσύνης και την δημιουργία ερευνητικών προγραμματιστικών μοντέλων.

Ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η **Python 3.12** και για την διαχείριση δεδομένων και αποτελεσμάτων την τεχνολογία **Python Jupyter Notebooks**, η τεχνολογία

αυτή παρέχει την δυνατότητα εκτέλεσης κατακερματισμένων κομματιών κώδικα καθως και την ενσωμάτωση κειμένου (markdown), κώδικα, γραφημάτων και αποτελεσμάτων εκτελέσιμου κώδικα σε μια κοινή και εύχρηστη διεπαφή χρήστη. Χρησιμοποιήθηκαν επίσης, βιβλιοθήκες της python οι οποίες περιγράφονται στον πίνακα 2.5.α. Η δημιουργία και εκτέλεση του κώδικα έγινε μέσω του περιβάλλοντος (IDE:Integrated Development Environment) **Visual Studio Code.**

Για την εμφάνιση του δικτύου διασύνδεσης χρηστών, σε μορφή ευανάγνωστη για το ανθρώπινο μάτι, χρησιμοποιήθηκε το εργαλείο **Gephi**, το οποίο είναι ένα πρόγραμμα ανάλυσης, εμφάνισης και μορφοποίησης γράφων πολλαπλών τύπων με δυνατότητες λειτουργίας σε πολύπλοκα γραφήματα, δηλαδή γραφήματα με πολλούς κόμβους και ακμές. Το εργαλείο **Zotero** χρησιμοποιήθηκε για την οργάνωση της βιβλιογραφίας.

Numpy	Για την δημιουργία πινάκων δεδομένων,την επεξεργασία τους καθώς και την εφαρμογή πράξεων γραμμικής άλγεβρας και άλλων μαθηματικών εργαλείων με αποδοτικό και γρήγορο τρόπο
Pandas	Για την αποθήκευση, εμφάνιση, επιλογη καθώς και την επεξεργασία δεδομένων
Matplotlib	Για την δημιουργία και εμφάνιση γραφικών χαρακτηριστικών όπως πίνακες και γραφήματα διαφόρων τύπων
NetworkX	Για την δημιουργία καθώς και την επεξεργασία των γράφων που δημιουργήθηκαν από τα δεδομένα εφαρμόζοντας τεχνικές της Θεωρίας Γραφημάτων
Scipy και SciKit-Learn	Για την χρήση έτοιμων εργαλείων υπολογισμου και μετρικών,καθώς και εφαρμογή έτοιμων μοντέλων ML
multiprocessing	Για παράλληλη αξιοποίηση των πυρήνων του επεξεργαστή κατά την εκτέλεση

Πίνακας 2.5.α: Βιβλιοθήκες Python που χρησιμοποιήθηκαν

Κεφάλαιο 3: Δημιουργία Μοντέλων

3.1: Δεδομένα

3.1.1: Παρουσίαση των δεδομένων

Τα δεδομένα που χρησιμοποιήθηκαν προέρχονται από την πλατφόρμα και βάση δεδομένων ταινιών IMDb (Internet Movie Database).

Εικόνα 3.1.1.α: IMDb
Πηγή: https://www.kaggle.com/datasets/hijest/genre-classification-dataset-imdb

Το σύνολο δεδομένων απαρτίζεται από κριτικές χρηστών σε διάφορες ταινίες. Κάθε εγγραφή έχει την εξής μορφή:

Id	User id	Item id	rating	date

Περιέχει δηλαδή το id του χρήστη που βαθμολόγησε την ταινία, το id της ταινίας που βαθμολογήθηκε από τον χρήστη, τον βαθμό πού έδωσε ο χρήστης στην ταινία (στο διάστημα από 1 έως 10), καθώς και την ημερομηνία της κριτικής αυτής. Το σύνολο δεδομένων περιέχει στην αρχική του μορφή 4.669.820 κριτικές, 1.499.238 διαφορετικούς χρήστες και 351.109 διαφορετικές ταινίες. Τα δεδομένα αυτά, είναι κατάλληλα για την εκπαίδευση και ανάλυση ενός RM (**Recommender Model**), περιέχοντας πολλαπλούς διαφορετικούς χρήστες και πολλαπλά αντικείμενα, πιο συγκεκριμένα για τεχνικές τύπου

συνεργατικού φιλτραρίσματος, συνεπώς το σύνολο δεδομένων ταιριάζει και στους σκοπούς αυτής της πτυχιακής εργασίας.

	user	item	rating	date
0	4592644	120884	10	2005-01-16
1	3174947	118688	3	2005-01-16
2	3780035	387887	8	2005-01-16
3	4592628	346491	1	2005-01-16
4	3174947	94721	8	2005-01-16
...	...	...	...	...
4669815	581842	107977	6	2005-01-16
4669816	3174947	103776	8	2005-01-16
4669817	4592639	107423	9	2005-01-16
4669818	4581944	102614	8	2005-01-16
4669819	1162550	325596	7	2005-01-16

4669820 rows × 4 columns

Εικόνα 3.1.1.β: Δεδομένα από IMDb σε μορφή pandas dataframe

Τα δεδομένα αυτά θα καθαριστουν και θα γίνει επιλογή του υποσυνόλου τους που θα χρησιμοποιηθεί στην αναπτυξη των μοντέλων. Στη συνέχεια θα εφαρμοστεί μια αρχική ανάλυση των δεδομένων , απο την οποία θα προκύψουν σημαντικά συμπεράσματα για τα χαρακτηριστικά τους καθώς και για την κατάλληλη αξιοποίηση από τα μοντέλα που θα υλοποιηθούν.

3.1.2: Καθαρισμός και Προετοιμασία των Δεδομένων

Το πρώτο βήμα είναι να εντοπιστούν και θα καθαριστούν τα διπλότυπα δεδομένα που μπορεί να υπάρχουν.

```
#find and remove duplicates and remove date column
duplicates=dataframe.duplicated().sum()
if (duplicates>0):
    print("Number of duplicate values:",duplicates)
    dataframe.drop_duplicates(inplace=True)
dataframe.drop(columns=["date"],inplace=True)

dataframe.to_pickle(pickle_file)

dataframe
```

[30] Python

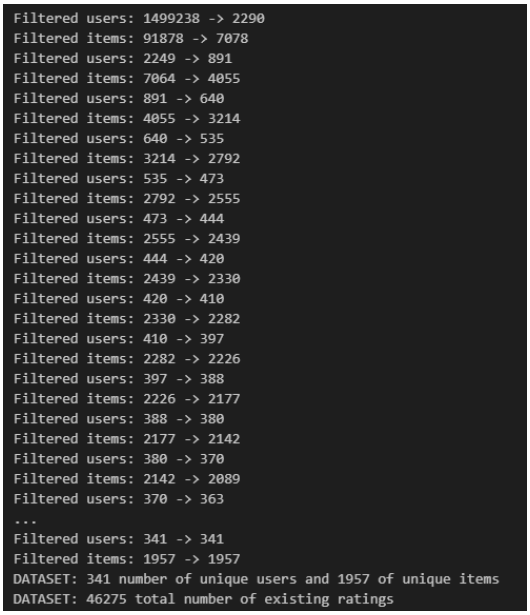
```
... Number of duplicate values: 9382
```

Εικόνα 3.1.2.α: Screenshot κώδικα αφαίρεσης διπλότυπων δεδομένων

Ο συνολικός αριθμός των δεδομένων είναι αρκετά μεγάλος, επιπλέον ένα μέρος των χρηστών καθως και των αντικειμένων συμμετέχουν σε πολύ λίγες αξιολογήσεις, έχουν έτσι περιορισμένες δυνατότητες για την εξαγωγή συμπερασμάτων. Επομένως εφαρμόζεται ένα φιλτράρισμα των χρηστών και τον αντικειμένων για να μειωθεί ο όγκος.Η επιλογή των ορίων έγινε πειραματικά με σκοπό την μείωση του όγκου δεδομένων διατηρώντας όμως χρήστες και αντικείμενα με πολλαπλές κριτικές διασφαλίζοντας την ύπαρξη αρκετής πληροφορίας στο τελικό σύνολο δεδομένων.

```python
minimum_user_ratings = 100
maximum_user_ratings = 300
minimum_item_ratings = 10

new_df = dataframe.copy()
# Initial filtering
def filter_users_and_items(df):
    # Filter users
    user_ratings_count = df.groupby("user")["rating"].count().sort_values(ascending=
                                    False).reset_index(name="ratings_num")
    filtered_users = user_ratings_count.loc[(user_ratings_count["ratings_num"] >= minimum_user_ratings) &
                                            (user_ratings_count["ratings_num"] <= maximum_user_ratings)]
    df = df.loc[df["user"].isin(filtered_users["user"])]
    print("Filtered users: {} -> {}".format(user_ratings_count.shape[0], filtered_users.shape[0]))

    # Filter items
    item_ratings_count = df.groupby("item")["rating"].count().sort_values(ascending=
                                    False).reset_index(name="users_rated")
    filtered_items = item_ratings_count.loc[item_ratings_count["users_rated"] >= minimum_item_ratings]
    df = df.loc[df["item"].isin(filtered_items["item"])]
    print("Filtered items: {} -> {}".format(item_ratings_count.shape[0], filtered_items.shape[0]))

    return df

# Iteratively filter users and items until no more changes
previous_shape = None
current_shape = new_df.shape

while previous_shape != current_shape:
    previous_shape = current_shape
    new_df = filter_users_and_items(   new_df)
    current_shape = new_df.shape

# Reset index and drop the old index column
final_df = new_df.reset_index(drop=True)

# Get final stats
users_num, items_num, ratings_num = get_stats(final_df)
```

Εικόνα 3.1.2.β: Screenshot κώδικα Φιλτραρίσματος των χρηστών και των αντικειμένων



Εικόνα 3.1.2.γ: Screenshot τελικού dataset

Καταλήγουμε, λοιπόν, σε 341 μοναδικούς χρήστες , 1957 μοναδικά αντικείμενα και 46275 κριτικές.

Στα φιλτραρισμένα δεδομένα, πραγματοποιείται ανανέωση των ids για τους χρήστες τα αντικείμενα καθώς και για τις εγγραφες, γενικά. Το τελικό σύνολο δεδομένων που προέκυψε, είναι έτοιμο για ανάλυση χρήση κατα την εκπαίδευση και αξιολόγηση των δύο μοντέλων.

```
# Get the unique users and items in the final dataframe along with the final
# number of ratings.
final_users = final_df["user"].unique()
final_items = final_df["item"].unique()
final_users_num = len(final_users)
final_items_num = len(final_items)
final_ratings_num = len(final_df)

# Report the final number of unique users and items in the dataset.
print("REDUCED DATASET: {0} number of unique users and {1} of unique items".format(final_users_num,final_items_num))
# Report the final  number of existing ratings in the dataset.
print("REDUCED DATASET: {} total number of existing ratings".format(final_ratings_num))
```

```
# We need to reset the users and items ids in order to be able to construct the
# networks of users and items. Users and Items ids should be consecutive integers
# in the [1...final_users_num] and [1...final_items_num].
# Initialy, we need to acquire the sorted versions of the user and item ids.
sorted_final_users = np.sort(final_users)
sorted_final_items = np.sort(final_items)


# Generate the dictionary of final users as a mapping of the following form:
# sorted_final_users --> [0...final_users_num-1]
final_users_dict = dict(zip(sorted_final_users,list(range(0,final_users_num))))
# Generate the dictionary of final items as a mapping of the following form:
# sorted_final_items --> [0...final_items_num-1]
final_items_dict = dict(zip(sorted_final_items,list(range(0,final_items_num))))
# Apply the previously defined dictionary-based maps on the users and item
# columns of the final dataframe.
final_df["user"] = final_df["user"].map(final_users_dict)
final_df["item"] = final_df["item"].map(final_items_dict)

final_df
```

Εικόνα 3.1.2.δ: Screenshot κώδικα ανανέωσης ids

	user	item	rating
0	0	312	7
1	31	751	9
2	5	597	6
3	5	590	6
4	0	513	6
...	...	...	...
46270	69	525	5
46271	56	700	10
46272	1	671	6
46273	0	464	6
46274	0	392	7

46196 rows × 3 columns

Εικόνα 3.1.2.ε: Screenshot τελικού dataframe με ανανεωμένα ids

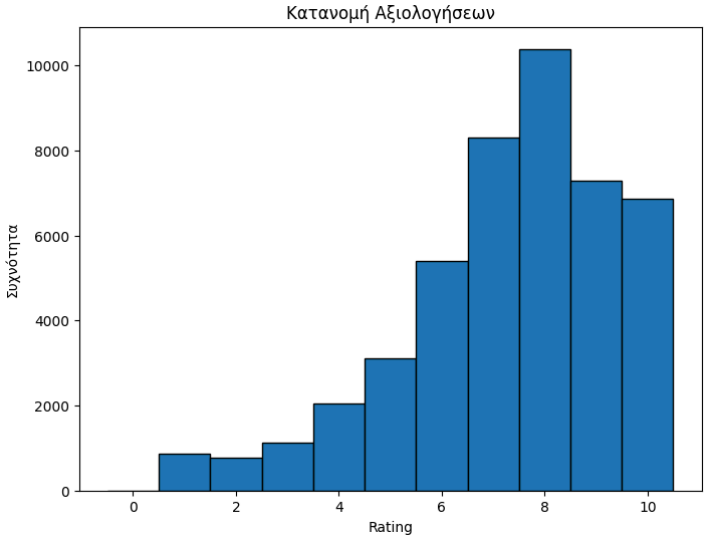
3.1.3: Ανάλυση δεδομένων και συμπεράσματα

Εφαρμόζοντας κάποιες βασικές τεχνικές ανάλυσης, προκύπτουν συμπεράσματα και κατευθύνσεις για την χρήση τους στην συνέχεια.

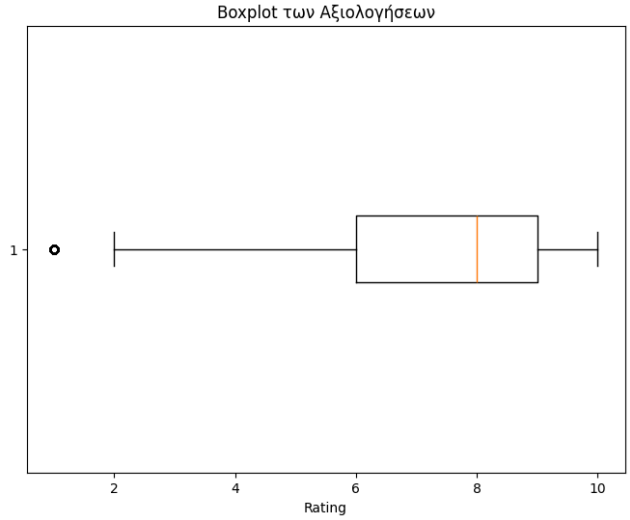
```
final_df["rating"].describe()
✓ 0.0s
count    45905.000000
mean         7.312254
std          2.087293
min          1.000000
25%          6.000000
50%          8.000000
75%          9.000000
max         10.000000
Name: rating, dtype: float64
```

Εικόνα 3.1.3.α: Βασικά στατιστικά για τις κριτικές

Μπορούμε να παρατηρήσουμε, λοιπόν, την κατανομή των αξιολογήσεων που έκαναν οι χρήστες. Οι περισσότερες αξιολογήσεις κυμαίνονται απο 6 εως 9.

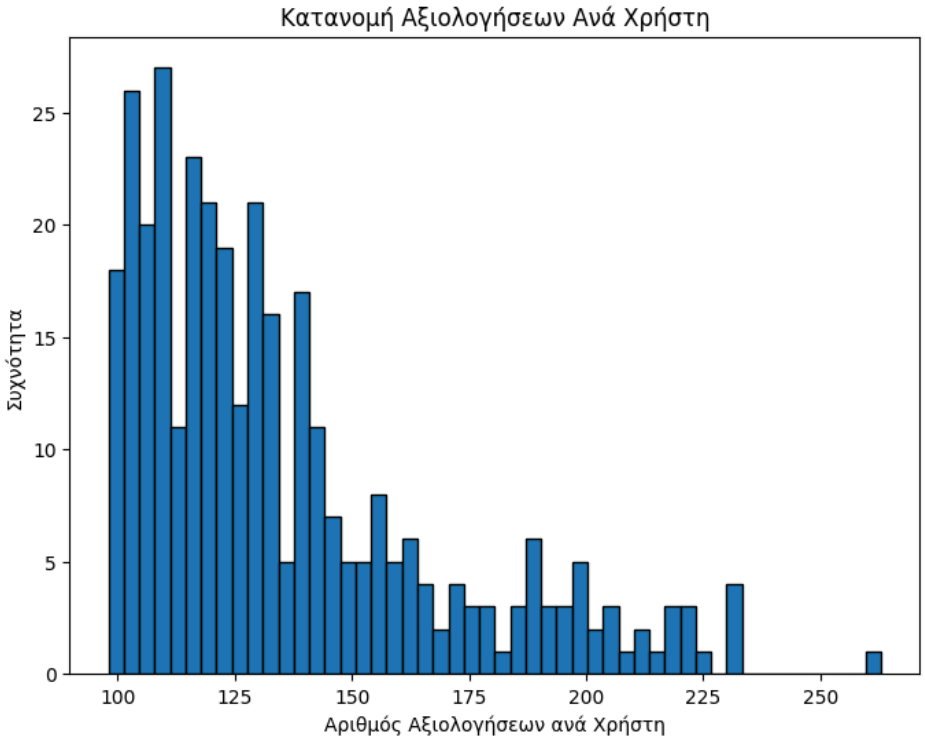


Διάγραμμα 3.1.3.α: Κατανομή Αξιολογήσεων



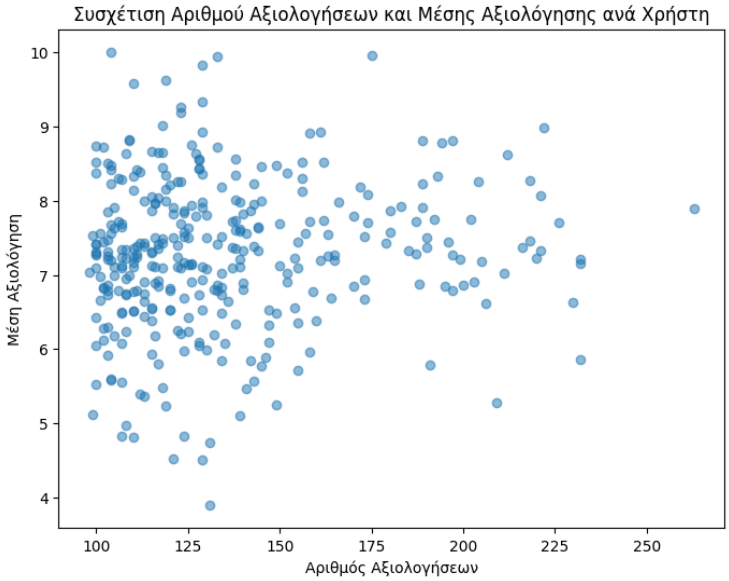
Διάγραμμα 3.1.3.β: Boxplot των Αξιολογήσεων

Οι περισσότεροι χρήστες μετά το φιλτράρισμα έχουν αξιολογήσει , περίπου, 100 με 150 αντικείμενα όπως φαίνεται στο παρακάτω διάγραμμα.



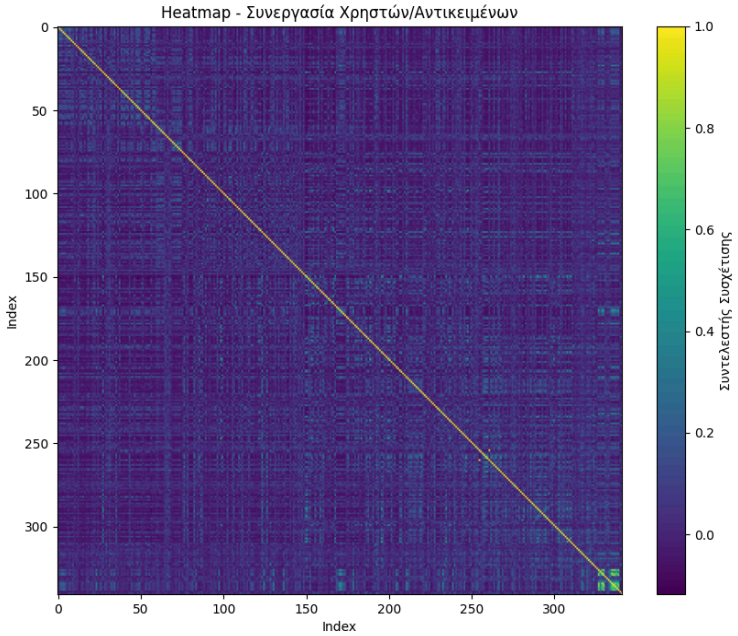
Διάγραμμα 3.1.3.γ: Κατανομή Αξιολογήσεων ανα Χρήστη

Παρατηρείται, επίσης, πως ο αριθμός τον αξιολογήσεων ανά χρήστη δεν σχετίζεται ιδιαίτερα με το αν οι αξιολογήσεις αυτές είναι ανα μέσο όρο υψηλότερες ή χαμηλότερες.



Διάγραμμα 3.1.3.δ: Διάσπαρτο Διάγραμμα (Scatter Plot) Συσχέτισης Αριθμού Αξιολογήσεων και Μέσης Αξιολόγησης ανα Χρήστη

Επιπλέον, δεν υπάρχει ξεκάθαρη συσχέτιση μεταξύ διαφορετικών χρηστών.



Διάγραμμα 3.1.3.ε: Heatmap Συνεργασίας Χρηστών/Αντικειμένων

3.2: GenRec1

3.2.1: Εκκίνηση GenRec1

Το πρώτο βήμα κατα την εκτέλεση του GenRec1 είναι ο διαχωρισμός του συνόλου δεδομένων, ο διαχωρισμός θα γίνει στο επίπεδο τον αντικειμένων, καθώς χρειαζόμαστε όλους τους χρήστες για την επεξεργασία του W. Τα αντικείμενα χωρίζονται σε δύο ομάδες, τα train αντικείμενα που θα χρησιμοποιηθούν κατά την εκπαίδευση, καθώς και τα test αντικείμενα που τα οποία θα αξιολογηθεί το τελικό αποτέλεσμα. Ιδιαίτερη σημασία κατέχει, η εύρεση των υποσυνόλων χρηστών που έχουν αξιολογήσει το κάθε αντικείμενο. Οι χρήστες αυτοί θα χωριστούν σε δύο κατηγορίες, σε γνωστούς και αγνωστους για κάθε αντικείμενο, και θα αποθηκευτούν στο λεξικό **item_split_dict** μαζί με τις κριτικές των γνωστών χρηστών για το κάθε αντικείμενο. Η διαδικασία αυτή εκτελείται στην συνάρτηση **precompute_item_split** κατα την εκκίνηση του GenRec1, προτού τα αντικείμενα χωριστούν σε train (validation) και test σύνολα.

```
79  #Splits the data into known and unknown ratings for each item and computes the known and unknown user sets
80  def precompute_item_split(split_ratio,dataframe):
81      final_items = final_df["item"].unique()
82      final_items_num = len(final_items)
83      sorted_final_items = np.sort(final_items)
84      final_items_dict = dict(zip(sorted_final_items,list(range(0,final_items_num))))
85      items_group_df=final_df.groupby("item")
86      item_users_dict={}
87      for item in final_items:
88          item_index=final_items_dict[item]
89          item_users=set(items_group_df.get_group(item_index)["user"])
90          item_users_dict[item_index]=item_users
91
92      item_split_dict = {}
93      cntr=0
94      for item, users in item_users_dict.items():
95          users = np.array(list(users))
96          np.random.shuffle(users)
97          split_idx = int(split_ratio * len(users))  # Fixed 50-50 split
98          known,unknown = (users[:split_idx], users[split_idx:])
99          itemdf = dataframe[dataframe['item'] == item]
100         ratings_that_exist = itemdf[itemdf['user'].isin(known)]['rating'].to_numpy()
101         Rtrue = itemdf[itemdf['user'].isin(unknown)]['rating'].to_numpy()
102         if len(Rtrue) !=len(unknown) or len(ratings_that_exist) != len(known):
103             print("size error while splitting", item, len(Rtrue), len(unknown), len(ratings_that_exist), len(known))
104             cntr+=1
105
106         item_split_dict[item] = (known,unknown, ratings_that_exist, Rtrue)
107     print("number of errors:",cntr)
108     final_users = final_df["user"].unique()
109     return item_split_dict, final_users.size
```

Εικόνα 3.2.1.α: Συναρτηση precompute item split

Το επόμενο βήμα κατα την εκκίνηση, είναι η δημιουργία του αρχικού πληθυσμού. Το βήμα αυτό μπορεί να υλοποιηθεί με διαφορετικούς τροπους, εξασφαλίζοντας πάντα ότι τα άτομα του πληθυσμού είναι συμμετρικοί πίνακες μεγέθους ***αριθμός χρηστων*** $\star$ ***αριθμός χρηστών***. Υλοποίησα αυτή την διαδικασία, δημιουργώντας έναν υβριδικό πληθυσμό, ο οποίος έχει επέλθει από την εφαρμογή των εξής μεθόδων:

Μεθόδοι
Non-negative Matrix Factorization
Spectral embedding using LaplaciansEigenvectors
RBF Kernels
SVD
Χαμηλοβαθμες μήτρες LRU (όπως χρησιμοποιούνται στον GenRec0)
Τυχαίες συμμετρικές μήτρες

Πίνακας 3.2.1.α: Μέθοδοι δημιουργίας υβριδικού πληθυσμού

Οι παραπάνω υλοποιήσεις εξασφαλίζουν την συμμετρία του πληθυσμού και φράζουν τις τιμές στο διαστημα τιμών $(0, 1) \in R$. Η υλοποίηση των μεθόδων βρίσκεται στο Παράρτημα Α.

```python
def create_hybrid_population(size, user_item_matrix):
    size_per_method = size // 6  # Split population evenly across methods
    populations = [
        create_nmf_population(size_per_method, user_item_matrix),
        create_spectral_population(size_per_method, user_item_matrix),
        create_svd_population(size_per_method, user_item_matrix),
        create_kernel_population(size_per_method, user_item_matrix),
        create_heuristic_symmetric_population(size_per_method, user_item_matrix),
        create_lowrank_population(size_per_method, user_item_matrix.shape[0])
    ]
    flat_population = [matrix for sublist in populations for matrix in sublist]
    return flat_population[:size]  # Trim to requested size
```

Εικόνα 3.2.1.β: Δημιουργία πληθυσμού

```python
def GenRec1(dataset,population_size, generations,sample_ratio=0.2,similarity_penalty=0.25,elitism=5,lamda_reg=0.01,split_ratio=0.5

    """Genetic algorithm for collaborative filtering with symmetric matrix factorization.Accepts a dataset and parameters for the
    initial_items,final_users_num = precompute_item_split(split_ratio,dataset)
    all_items=list(initial_items.keys())
    user_similarity=compute_user_similarity_matrix(dataset)

    # Precompute fixed validation subset once (e.g., 20% of items)
    fixed_validation_size = int(sample_ratio * len(all_items))
    fixed_validation_items = random.sample(all_items, fixed_validation_size)  # Fixed seed for reproducibility
    current_items_subset = {k: initial_items[k] for k in fixed_validation_items}  # Use this for all generations
    test_items = set(all_items) - set(fixed_validation_items)
    test_subset = {k: initial_items[k] for k in test_items}

    populations = {}
    initial_pop=create_hybrid_population(population_size,user_similarity)
    populations["initial"]=initial_pop
    pop = initial_pop
    graph_data=[]
    #initial_gen_items=item_selector(initial_items,10)
```

Εικόνα 3.2.1.γ: Βήματα κατα την εκκίνηση του GenRec1

3.2.2: Κατα την εκτέλεση της κάθε γενιάς του GenRec1

Ο αλγόριθμος είναι έτοιμος να ξεκινήσει την εξελικτική διαδικασία , αξιολογώντας τον αρχικό πληθυσμό. Για κάθε άτομο του πληθυσμού υπολογίζεται ο δείκτης καταλληλότητας (fitness score). Αυτό πετυχαίνεται, μέσω της συνάρτησης **evaluate_individual**, ώπου για κάθε αντικείμενο του train set υπολογίζονται οι υπομήτρες του W individual που εξετάζεται και προχωράει ο υπολογισμός του σφάλματος για κάθε αντικείμενο. Το σφάλμα δεν είναι άλλο από το MSE των αποκλίσεων, μεταξύ γνωστών αξιολογήσεων και προβλέψεων της συνάρτησης (1). Ο υπολογισμος της (1) υλοποιήθηκε μέσω του εργαλείου minres (της βιβλιοθήκης scipy) για την επίλυση γραμμικών συστημάτων στην συνάρτηση **predict_ratings3**, οι τελικές προβλέψεις κανονικοποιούνται στο διακριτό διάστημα $[0, 10] \in Z$ και υπολογίζεται το MSE.

```
66  def predict_ratings3(Wu, Wl, Rmi, Rtrue, lambda_reg=0.01):
67      identity = np.eye(Wu.shape[0])
68      A = identity - Wu + lambda_reg * identity
69      b = Wl @ Rmi
70      try:
71          Ri_pred = minres(A, b, maxiter=200, rtol=1e-3)[0]
72      except:
73          Ri_pred = np.linalg.lstsq(A, b, rcond=None)[0]
74      Ri_pred = quick_norm10(Ri_pred)
75      Ri_error = np.mean((Ri_pred - Rtrue) ** 2)  # Vectorized MSE
76      return Ri_error
```

Εικόνα 3.2.2.α: Υπολογισμός σφάλματος

Έχοντας καταλήξει στο σφάλμα για κάθε αντικείμενο, υπολογίζουμε τον μέσο όρο των σφαλμάτων, για να αξιολογήσουμε την καταλληλότητα του individual που εξετάζεται. Η διαδικασία επαναλαμβάνεται για κάθε άτομο του πληθυσμού.

```
111 #function to evaluate the fitness of an individual
112 def evaluate_individual(W,item_subset,lamda_reg=0.01):
113     total_error=0.0
114     count=0
115     error_sum1 = 0
116     error_sum2 = 0
117     for item,(known,unknown,ratings_that_exist, Rtrue) in item_subset.items():
118         W_unknown = W[np.ix_(unknown, unknown)]
119         W_known_to_unkonown = W[np.ix_(unknown, known)]
120         if ratings_that_exist.size != W_known_to_unkonown.shape[1]:
121             print(f"Mismatch: ratings_that_exist size {ratings_that_exist.size}, W_known_to_unknown columns {W_known_to_unkonown.shape[1]}")
122             ratings_that_exist = ratings_that_exist[:W_known_to_unkonown.shape[1]]
123             #print(ratings_that_exist)
124             #print(W_known_to_unkonown)
125             #break
126             error_sum1+=1
127 
128         if Rtrue.size != W_unknown.shape[0]:
129             print(f"Mismatch: Rtrue size {Rtrue.size}, W_unknown rows {W_unknown.shape[0]}")
130             Rtrue = Rtrue[:W_unknown.shape[0]]
131             error_sum2+=1
132         error = predict_ratings3(W_unknown, W_known_to_unkonown, ratings_that_exist, Rtrue,lambda_reg=lamda_reg)
133         total_error += error
134         count+=1
135     return total_error/count if count > 0 else np.inf
```

Εικόνα 3.2.2.β: Υπολογισμος fitness score με την συνάρτηση evaluate_individual

Ο τελικός δείκτης καταλληλότητας, υπολογίζεται συμψηφίζοντας έναν αρνητικό συντελεστή ομοιότητας, “τιμωρώντας” τις πολύ όμοιες λύσεις και τα άτομα του πληθυσμού ταξινομούνται με βάση αυτόν τον δείκτη.

```python
with Pool(processes=os.cpu_count() -1) as pool:

    for generation in range(1,generations):
        #randomly select items for evaluation
        #selected_items=random.sample(all_items, int(sample_ratio * len(all_items)))
        #current_items_subset = {k: initial_items[k] for k in selected_items}

        args = [(ind, current_items_subset,lamda_reg) for ind in pop]
        fitness={}
        fitness_vector=pool.starmap(evaluate_individual, args)
        fitness = {tuple(ind.flatten()): err for ind, err in zip(pop, fitness_vector)}
        diversity = calculate_diversity(pop)

        # Combine fitness and diversity (weighted sum)
        # Weight for diversity penalty=similarit_penalty
        combined_fitness = {
            k: fitness[k] + similarity_penalty * diversity[k]  # Penalize similarity
            for k in fitness
        }


        sorted_individuals = sorted(fitness.keys(), key=lambda x: combined_fitness[x])
        #elitism keeping top n individuals
```

Εικόνα 3.2.2.γ: Υπολογισμός fitness score για τον πληθυσμό

```python
#penalize too similar encouraging diversity
def calculate_diversity(population):
    diversity_penalty = {}
    for i, ind1 in enumerate(population):
        similarity = 0
        for ind2 in population:
            if np.array_equal(ind1, ind2):
                continue
            # Compute cosine similarity between flattened matrices
            similarity += np.dot(ind1.flatten(), ind2.flatten()) / (np.linalg.norm(ind1) * np.linalg.norm(ind2))
        diversity_penalty[tuple(ind1.flatten())] = similarity
    return diversity_penalty
```

Εικόνα 3.2.2.δ: Diversity penalty

Στην συνέχεια, ο γενετικός αλγόριθμος προχωράει στο επόμενο βήμα της δημιουργίας νέου πληθυσμου. Υλοποιείται ένας μηχανισμός ελιτισμού (elitism) διατηρώντας λίγους από τους καλύτερους χρήστες για την επομενη γενιά. Πραγματοποιείται ζευγάρωμα (crossover) για την δημιουργία απογόνων για τον νέο πληθυσμό. Επιλέγονται τυχαία χρωμοσώματα

του κάθε γονέα για να προχωρήσουν στην επομενη γενιά και διατηρείται η συμμετρία μέσω της συνάρτησης uniform crossover[23].

```python
#uniform crossover ensuring symmetric offspring
def uniform_crossover(parent1, parent2):
    # Allow asymmetric exploration by not enforcing symmetry during crossover
    mask = np.random.rand(*parent1.shape) < 0.5  # Random mask without symmetry
    offspring1 = parent1 * mask + parent2 * (1 - mask)
    offspring2 = parent2 * mask + parent1 * (1 - mask)
    # Symmetrize offspring after crossover
    offspring1 = (offspring1 + offspring1.T) / 2
    offspring2 = (offspring2 + offspring2.T) / 2
    return offspring1, offspring2
```

Εικόνα 3.2.2.ε: Uniform crossover, συμμετρικό ζευγάρωμα γονέων

Έπειτα εφαρμόζεται τυχαία μετάλλαξη (mutation) σε κάποια γονίδια των ατόμων του νέου πληθυσμού, διατηρώντας πάντα τη συμμετρία.

```python
def mutation2(individual, mutation_rate=0.1, scale=0.5):  # Reduced from 0.5 to 0.1
    # Asymmetric noise during evolution
    noise = np.random.normal(scale=scale, size=individual.shape)
    mask = np.random.rand(*individual.shape) < mutation_rate
    mutated = individual + mask * noise
    # Symmetrize only after mutation
    mutated = (mutated + mutated.T) / 2  # Ensure symmetry post-mutation
    mutated = np.clip(mutated, 0.001, 0.999)
    return mutated
```

Εικόνα 3.2.2.στ: Μετάλλαξη μέσω συνάρτησης mutation 2

Σε αυτή την φάση ο νέος πληθυσμός για την επόμενη γένια έχει δημιουργηθεί.

Η διαδικασία που αναφέρθηκε, τρέχει διαδοχικά για έναν προκαθορισμένο αριθμό γενεών. Για λόγους χρονικής απόδοσης, αξιοποιείται παράλληλος υπολογισμός μέσω της βιβλιοθήκης Pool της python, συγκεκριμένα ο υπολογισμός της συνάρτησης evaluate_individual, τρέχει παράλληλα στα threads του μηχανήματος κατά τη εκτέλεση.

```
with Pool(processes=os.cpu_count() -1) as pool:

    for generation in range(1,generations):
        #randomly select items for evaluation
        #selected_items=random.sample(all_items, int(sample_ratio * len(all_items)))
        #current_items_subset = {k: initial_items[k] for k in selected_items}

        args = [(ind, current_items_subset,lamda_reg) for ind in pop]
        fitness={}
        fitness_vector=pool.starmap(evaluate_individual, args)
        fitness = {tuple(ind.flatten()): err for ind, err in zip(pop, fitness_vector)}
        diversity = calculate_diversity(pop)

        # Combine fitness and diversity (weighted sum)
        # Weight for diversity penalty=similarit_penalty
        combined_fitness = {
            k: fitness[k] + similarity_penalty * diversity[k]  # Penalize similarity
            for k in fitness
        }


        sorted_individuals = sorted(fitness.keys(), key=lambda x: combined_fitness[x])
        #elitism keeping top n individuals

        new_pop = [np.array(ind).reshape(final_users_num, final_users_num)
                   for ind in sorted_individuals[:elitism]]
        while len(new_pop) < population_size:
            parent1, parent2 = random.choices(pop[:population_size], k=2)  # Tournament selection
            offspring1,offspring2 = uniform_crossover(parent1, parent2)
            offspring1 = mutation1(offspring1,mutation_rate,scale)#with parameters
            offspring2 = mutation1(offspring2,mutation_rate,scale)#with parameters
            new_pop.extend((offspring1,offspring2))

        pop = new_pop[:population_size]
        #best individual from last gen is also  preserved because of elitism
        best_individual=pop[0]

        populations[generation]=pop
        best_fitness=min(np.array(list(fitness.values())))
        avg_fitness=np.average(np.array(list(fitness.values())))
        graph_data.append([generation, avg_fitness])

        print(f"Generation {generation} complete,average fitness:, {avg_fitness}")
```

Εικόνα 3.2.2.ζ: Κώδικας που εκτελείται ανα γενιά

3.2.3: Κατα το τέλος της διαδικασίας του GenRec1

Αφού έχουν ολοκληρωθεί όλες οι γενιές εκπαίδευσης, υπολογίζεται το **test_score** για τον καλύτερο individual που εξελίχθηκε. Παρουσιάζονται διαγράμματα για την πορεία του μέσου fitness της κάθε γενιάς καθώς και η μορφή του τελικού W ως heatmap. Τα τελικα αποτελέσματα αποθηκεύονται.

```
plt.colorbar(label="Συντελεστής Συσχέτισης")
plt.title("Heatmap - Συσχέτιση χρηστών στον W")
plt.xlabel("Index")
plt.ylabel("Index")
plt.show()
folder = "figures"  # Change this to your desired folder
os.makedirs(folder, exist_ok=True)  # Create folder if it doesn't exist
file_path = os.path.join(folder, f"best individual{TIMESTAMP}.png")
plt.savefig(file_path)



#plotting fitness per gen
plt.figure(figsize=(8, 5))
plt.plot(*zip(*graph_data), marker='o', linestyle='-', color='b', label="Avg Fitness")
plt.ylim(0, max([item[1] for item in graph_data]))
plt.xlabel("Generation")
plt.ylabel("Average Fitness")
plt.title("Generation vs Average Fitness")
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
folder = "figures"  # Change this to your desired folder
os.makedirs(folder, exist_ok=True)  # Create folder if it doesn't exist
file_path = os.path.join(folder, f"Fitness_plot{TIMESTAMP}.png")
plt.savefig(file_path)

test_score = evaluate_individual(best_individual, test_subset)
print("Last generation complete, best fitness:{0}avg_fitness:{1}test fitness:{2}".format(best_fitness,avg_fitness,test_score))
return best_individual,best_fitness,avg_fitness,test_score
```

Εικόνα 3.2.3.α: Κώδικας της διαδικασίας κατα την ολοκλήρωση

3.3 GenRec0

3.3.1: Εκκίνηση GenRec0

Κατά την εκκίνηση του GenRec0, εκτελείται η **precompute_item_split** με τον ίδιο τρόπο όπως και στον GenRec1 χωρίζοντας, δηλαδή, το σύνολο δεδομένων και υπολογίζοντας τα υποσύνολα των χρηστων που αξιολόγησαν ή δεν αξιολόγησαν το κάθε αντικείμενο, καθώς και ο διαχωρισμός του συνόλου δεδομένων σε αυτά που θα χρησιμοποιηθούν στην εκπαίδευση και στον έλεγχο. Δημιουργείται ο αρχικός πληθυσμός αποτελουμενος απο μήτρες διάστασης $\text{αριθμος χρηστών} \star RANK$.

Η επιλογή του RANK είναι σημαντική, καθώς καθορίζει πόσες υπερπαραμέτρους (embeddings) θα χρησιμοποιήσει ο αλγόριθμος για κάθε χρήστη. Μεγαλύτερο RANK σημαίνει και περισσότερη πληροφορία, αλλά μεγαλύτερη υπολογιστική πολυπλοκότητα. Ενώ μικρότερο RANK, καθιστά πιο γρήγορο αλγόριθμο, αλλά λιγότερη πληροφορία για κάθε χρήστη.

```
53  def create_initial_population0(size, n_users, rank=RANK):
54      return [np.random.randn(n_users, rank) for _ in range(size)]
```

Εικόνα 3.3.1.α: Δημιουργία πληθυσμού

Μετα την δημιουργία του πληθυσμου η επαναληπτική διαδικασία μπορεί να ξεκινήσει.

3.3.2: Κατα την εκτέλεση της κάθε γενιάς του GenRec0

Για την αξιολόγηση των ατόμων στον πληθυσμό ο GenRec0 χρησιμοποιεί την **evaluate_individual0**. Οι προβλέψεις των αξιολογήσεων υπολογίζονται μέσω της συνάρτησης predict_ratings0, εφαρμόζοντας τον τύπο (2) και συγκρίνονται με τις πραγματικές τιμες για να προκύψει το MSE του κάθε αντικειμένου. Ο μέσος όρος των σφαλμάτων αυτών είναι βαθμός καταλληλότητας για το κάθε άτομο, ομοίως με τον GenRec1.

```
56  def predict_ratings0(U, known_users, unknown_users, R_known):
57      # Compute affinity between unknown and known users
58      W_subset = U[unknown_users] @ U[known_users].T
59      # Normalize rows to sum to 1 (softmax)
60      W_subset = np.exp(W_subset) / np.sum(np.exp(W_subset), axis=1, keepdims=True)
61      # Predict as weighted average of known ratings
62      return W_subset @ R_known
```

Εικόνα 3.3.2.α: Εφαρμογή της συνάρτησης (2) μεσω predict_ratings0

```
65  def evaluate_individual0(U, item_subset):
66      total_error = 0.0
67      count = 0
68      for item, (known, unknown, R_known, R_true) in item_subset.items():
69          # Skip items with no known/unknown users
70          if len(known) == 0 or len(unknown) == 0:
71              continue
72          # Predict and calculate error
73          R_pred = predict_ratings0(U, known, unknown, R_known)
74          error = np.mean((R_pred - R_true) ** 2)
75          total_error += error
76          count += 1
77      return total_error / count if count > 0 else np.inf
78
```

Εικόνα 3.3.2.β: Υπολογισμός μέσου τετραγωνικού σφάλματος

Αφού έχει υπολογιστεί ο βαθμός καταλληλότητας κάθε ατόμου στον πληθυσμό με την συνάρτηση καταλληλότητας που ορίσαμε, τα άτομα ταξινομούνται,, και οι κορυφαιες λυσεις προστίθενται στον πληθυσμό της επόμενης γενιάς, εφαρμοζοντας elitism.

Στην συνέχεια, δημιουργούνται νέα άτομα μέσω μιας συνάρτησης ζευγαρώματος (crossover) και της εφαρμογής μιας συνάρτησης μετάλλαξης (mutation).

```python
def crossover0(parent1, parent2):
    # Column-wise crossover
    mask = np.random.rand(RANK) < 0.5
    child1 = parent1 * mask + parent2 * (~mask)
    child2 = parent2 * mask + parent1 * (~mask)
    return child1, child2

def mutation0(individual, mutation_rate=0.1, scale=0.1):
    noise = np.random.normal(scale=scale, size=individual.shape)
    mask = np.random.rand(*individual.shape) < mutation_rate
    return individual + mask * noise

```

Εικόνα 3.3.2.γ: Συναρτήσεις crossover και mutation στον GenRec0

Εμφανίζεται η μέση απόδοση του πληθυσμού, καθώς και το καλύτερο άτομο που αξιολογήθηκε και ο νέος πληθυσμός αποθηκευεται για την επόμενη γενιά. Η διαδικασία σταματάει όταν έχει συμπληρωθεί ένας προκαθορισμένος αριθμός γενιών.

```python
# Evolution loop
with Pool(processes=os.cpu_count()-1) as pool:
    for gen in range(generations):
        # Evaluate fitness
        args = [(ind, validation_subset) for ind in population]
        fitness_results = pool.starmap(evaluate_individual0, args)
        fitness = {tuple(ind.flatten()): score for ind, score in zip(population, fitness_results)}

        # Selection
        sorted_pop = [ind for _, ind in sorted(zip(fitness_results, population), key=lambda x: x[0])]

        # New population
        new_pop = sorted_pop[:elitism]  # Keep top elites

        # Breed remaining population
        while len(new_pop) < population_size:
            parents = random.choices(sorted_pop[:population_size//2], k=2)
            child1, child2 = crossover0(parents[0], parents[1])
            new_pop.extend([
                mutation0(child1, mutation_rate, scale),
                mutation0(child2, mutation_rate, scale)
            ])

        population = new_pop[:population_size]
        best_embeding = population[0]# Get the individual with the lowest fitness value
        # Logging
        best_fitness = min(fitness.values())
        avg_fitness = np.mean(list(fitness.values()))
        graph_data.append((gen, avg_fitness))
        print(f"Gen {gen}: Best={best_fitness:.2f}, Avg={avg_fitness:.2f}")
```

Εικόνα 3.3.2.δ: Αναλυτικά βήματα ανα γενιά στον GenRec0

3.3.3: Κατα το τέλος της διαδικασίας του GenRec0

Μετά την εκτέλεση της τελευταίας γενιάς, επιλέγεται ο καλύτερος πίνακας U που εξελίχθηκε, από αυτόν δημιουργείται μια εκτίμηση του πίνακα W πολλαπλασιάζοντας με τον ανάστροφο:

```
best_individual=best_embeding@best_embeding.T
```

Εμφανίζονται κάποια διαγράμματα και επιστρέφεται η μέση τιμή των αποδόσεων για κάθε individual ο καλύτερος individual μαζι με την απόδοση του καθώς και το test score για αυτόν.

```
#ploting avg fitness
plt.figure(figsize=(8, 5))
plt.plot(*zip(*graph_data), marker='o', linestyle='-', color='b', label="Avg Fitness")
plt.ylim(0, max([item[1] for item in graph_data]))
plt.xlabel("Generation")
plt.ylabel("Average Fitness")
plt.title("Generation vs Average Fitness")
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()

folder = "figures"  # Change this to your desired folder
os.makedirs(folder, exist_ok=True)  # Create folder if it doesn't exist
file_path = os.path.join(folder, f"Fitness_plot{TIMESTAMP}.png")
plt.savefig(file_path)

test_score = evaluate_individual0(population[0], test_subset)
print("Last generation complete, best fitness:{0}avg_fitness:{1}test score:{2}".format(best_fitness,avg_fitness,test_score))
return best_individual,best_fitness, avg_fitness, test_score
```

Εικόνα 3.3.3.α: Τερματισμός GenRec0

3.4: Εκτέλεση και Αποθήκευση

Για την ομαλή εκτέλεση, καθώς και την αποθήκευση των αποτελεσμάτων μετά την ολοκλήρωση ευθύνεται το παρακάτω χωρίο του κώδικα στα αρχεία GenRec1 και GenRec0, αντίστοιχα:

```python
if __name__ == '__main__':
    print("Script started")
    datafolder = "datafiles"
    pickle_file = os.path.normpath(os.path.join(datafolder, "final_df.pkl"))

    try:
        final_df = pd.read_pickle(pickle_file)
        print("Data loaded successfully")
    except Exception as e:
        print(f"Error loading data: {e}")
        exit()


    try:
        unique_items = final_df["item"].unique()
        unique_users= final_df["user"].unique()
        print(f"Unique items: {unique_items.shape}")
        users_that_rated = {item: final_df[final_df["item"] == item]['user'].values for item in unique_items}
        print("Users that rated initialized successfully")
    except Exception as e:
        print(f"Failed to initialize users_that_rated: {e}")

    param_combinations =[
        {
            "population_size": 30,
            "generations": 150,
            "sample_ratio": 0.8,
            "similarity_penalty": 0.4,
            "elitism":2,
            "lamda_reg":0.001,
            "split_ratio": 0.8,
            "noise_scale": 0.4,
            "mutation_rate": 0.005,
            "scale":0.5
        }
    ]




    for params in param_combinations:
        print(f"Testing parameters: {params}")
        print("Running GenRec1")
        best_individual,best_fitness, avg_fitness,test_score = GenRec1(final_df, **params)
        cluster_graph(best_individual)
        log_results(params, best_fitness, avg_fitness,test_score)
```

Εικόνα 3.4.α: Παράδειγμα script υπευθύνου για την εκτέλεση GenRec1

Τα δεδομένα φορτώνονται και αφού έχουν καθοριστεί οι παράμετροι το μοντέλο εκτελείται. Υπάρχει δυνατότητα πολλαπλών εκτελέσεων με διαφορετικές παραμέτρους. Μετά την κάθε εκτέλεση τα αποτελέσματα αποθηκεύονται σε κατάλληλο αρχείο.Συγκεκριμένα αποθηκεύονται: οι παράμετροι που χρησιμοποιήθηκαν, η καλύτερη απόδοση

(best_fitness),η μέση απόδοση (average_fitness), το test_score, καθώς και η χρονική διάρκεια της κάθε εκτέλεση. Αυτά τα δεδομένα, αποθηκεύονται σε csv αρχείο που μπορεί να χρησιμοποιηθεί για περαιτέρω ανάλυση των αποτελεσμάτων. Επίσης, αποθηκεύεται η καλύτερη λύση που ανέδειξε το μοντέλο με την οποία δημιουργείται γράφος διασύνδεσης χρηστών. Στον γράφο αυτό εφαρμόζεται ομαδοποίηση(clustering) των χρηστών για εξαγωγή επιπλέον συμπερασμάτων απο την μορφή της τελικής λύσης, χρησιμοποιώντας την συνάρτηση cluster_graph.

```python
def cluster_graph(W,threshold=0.2):
    # Approximate kernel embeddings
    W_thresholded = np.where(W > threshold, W, 0)
    G = nx.from_numpy_array(W_thresholded)

    # Add node metadata (example: cluster labels)
    clusters = SpectralClustering(n_clusters=5,affinity='precomputed').fit_predict(W_thresholded)  # Optional
    nx.set_node_attributes(G, dict(enumerate(clusters)), 'cluster')


    folder = "figures"  # Change this to your desired folder
    os.makedirs(folder, exist_ok=True)  # Create folder if it doesn't exist
    file_path = os.path.join(folder, f"GRAPH{timestamp}.gexf")

    nx.write_gexf(G,file_path)
```

Εικόνα 3.4.β: Δημιουργία γράφου και ομαδοποίηση των χρηστών

Κεφάλαιο 4: Εκτέλεση και Αποτελέσματα

4.1: Παραμετροποίηση

Ιδιαίτερη σημασία για την βελτιστοποίηση της λειτουργίας του κάθε γενετικού αλγορίθμου κατα την εκτέλεση, έχει η βέλτιστη επιλογή παραμέτρων.

Ανάλυση των παραμέτρων κατα την εκκίνηση

ΠΑΡΑΜΕΤΡΟΣ	ΕΞΗΓΗΣΗ
`population_size`	Μέγεθος πληθυσμού(GenRec1 και GenRec0)
`generations`	Αριθμός γενιών(GenRec1 και GenRec0)
`sample_ratio`	Αναλογία των αντικειμένων που χρησιμοποιείται για εκπαίδευση(GenRec1 και GenRec0)
`similarity_penalty`	Ποινή για την ομοιότητα πανομοιότυπων χρωμοσωμάτων(μόνο GenRec1)
`elitism`	Αριθμός ατόμων που διατηρούνται στην επόμενη γενιά μέσω ελιτισμού(GenRec1 και GenRec0)
`lamda_reg`	Συντελεστής στην συνάρτηση (1)(μόνο GenRec1)
`split_ratio`	Διαχωρισμός του υποσυνόλου των χρηστων για κάθε αντικείμενο σε γνωστούς και αγνώστους(GenRec1 και GenRec0)
`noise_scale`	Κλίμακα θορύβου για την δημιουργία ατόμων(δεν χρησιμοποιείται)(μόνο GenRec1)
`mutation_rate`	Βαθμός μετάλλαξης(GenRec1 και GenRec0)
`scale`	Κλίμακα μετάλλαξης(GenRec1 και GenRec0)

`RANK`	Βαθμός των LRUs(Μόνο GenRec0)

Πίνακας 4.1.α: Πίνακας παραμέτρων κατά την εκκίνηση

Εκτελώντας πειραματικά τα δύο μοντέλα για διάφορους συνδυασμούς παραμέτρων, κατέληξα στις εξής συνδυασμούς για το GenRec1 και GenRec0 οι οποίοι επιδρούν πιο θετικά στα αποτελέσματα. Οι εξής συνδυασμοί είναι αυτοί που είχαν ώς αποτέλεσμα το χαμηλότερο σφάλμα κατα την εκτέλεση για τους δύο αλγορίθμους αντίστοιχα.

ΠΑΡΑΜΕΤΡΟΣ	GenRec1	GenRec0
`sample_ratio`	0.3	0.6
`similarity_penalty`	0.2	-
`elitism`	12	4
`lamda_reg`	0.1	-
`split_ratio`	0.8	0.8
`noise_scale`	0.9	-
`mutation_rate`	0.3	0.03
`scale`	0.1	0.1
`RANK`	-	30

Πίνακας 4.1.β: Βέλτιστοι παράμετροι για GenRec1 και GenRec0

Η διαδικασία αυτή της ανάδειξης βέλτιστων παραμέτρων είναι σημαντική για την ανάπτυξη κάθε ΓΑ, καθώς οι παράμετροι όπως ο βαθμός μετάλλαξης επηρεάζουν δραστικά τον τροπο που ο αλγόριθμος εξελίσει νέες λύσεις στο πρόβλημα.

4.2: GenRec1

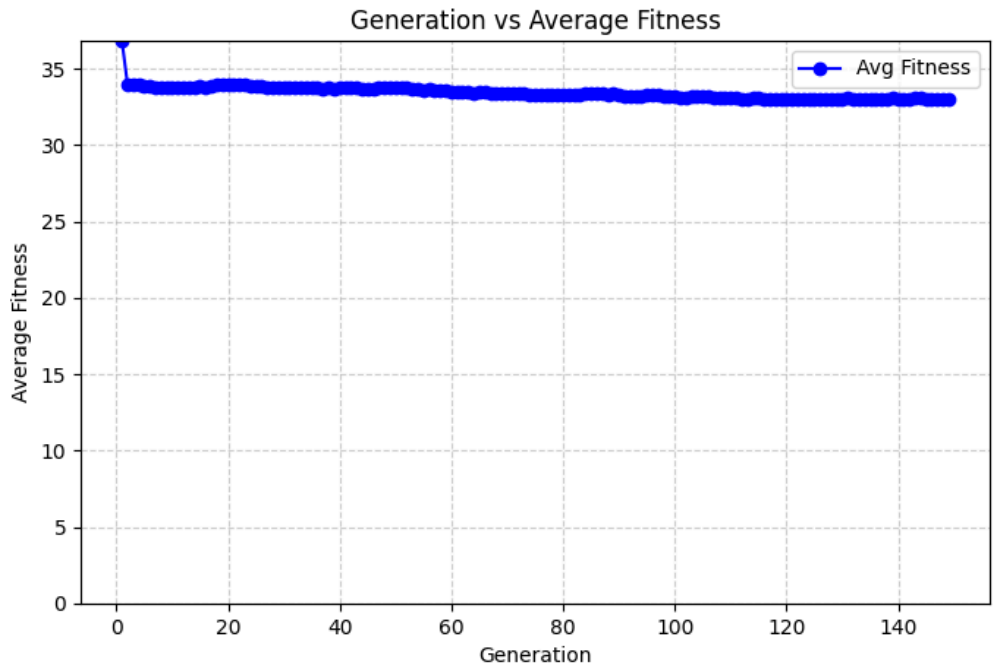
Εκτελωντας το GenRec1 για διαφορετικές παραμέτρους παρατηρούμε ότι το Μέσο Τετραγωνικό Σφάλμα δεν μειώνεται δραστικά κάτω από το κατώφλι του 29(RMSE:5.3) ανεξαρτήτως από τον αριθμό των γενιών και την αλλαγή άλλων παραμέτρων του GA.

	population_size	generations	sample_ratio	similarity_penalty	elitism	lamda_reg	split_ratio	noise_scale	mutation_rate	scale	best_fitness	avg_fitness	test_score	timestamp	duration
31	50	500	0.3	0.20	6	0.100	0.80	0.20	0.100	0.1	29.979104	30.558290	NaN	2025-02-03 15:46:57	NaN
139	20	150	0.3	0.25	5	0.010	0.80	0.20	0.010	0.1	30.452496	30.608713	NaN	2025-01-31 01:12:21	NaN
30	50	500	0.3	0.25	10	0.010	0.80	0.20	0.010	0.1	30.488510	31.183906	NaN	2025-02-03 15:03:40	NaN
22	25	100	0.3	0.20	12	0.100	0.80	0.90	0.300	0.1	31.077278	31.479451	NaN	2025-01-31 00:56:42	NaN
20	50	100	0.3	0.20	12	0.100	0.80	0.90	0.200	0.1	31.272781	32.354477	NaN	2025-01-31 00:51:04	NaN

Εικόνα 4.2.α: Αποτελέσματα GenRec1 για διαφορετικές παραμέτρους

Η μέση καταλληλότητα κατα την εκτέλεση μένει σχεδόν σταθερή από γενιά σε γενιά, παρόλο που παρατηρείται μια σταθερή μικρή μείωση του σφάλματος .

Η καταλληλότητα τώρα κατα τον έλεγχο με την χρήση νέων δεδομένων(αντικείμενα με τα οποία δεν έχει εξελιχθεί ο αλγόριθμος) , που προσμετράται με την μεταβλητή test_score εμφανίζει αντίστοιχα δυσμενή αποτελέσματα.Η τιμή της στις καλύτερες εκτελέσεις δεν ξεπέρασε το MSE=33(RMSE=5,7).



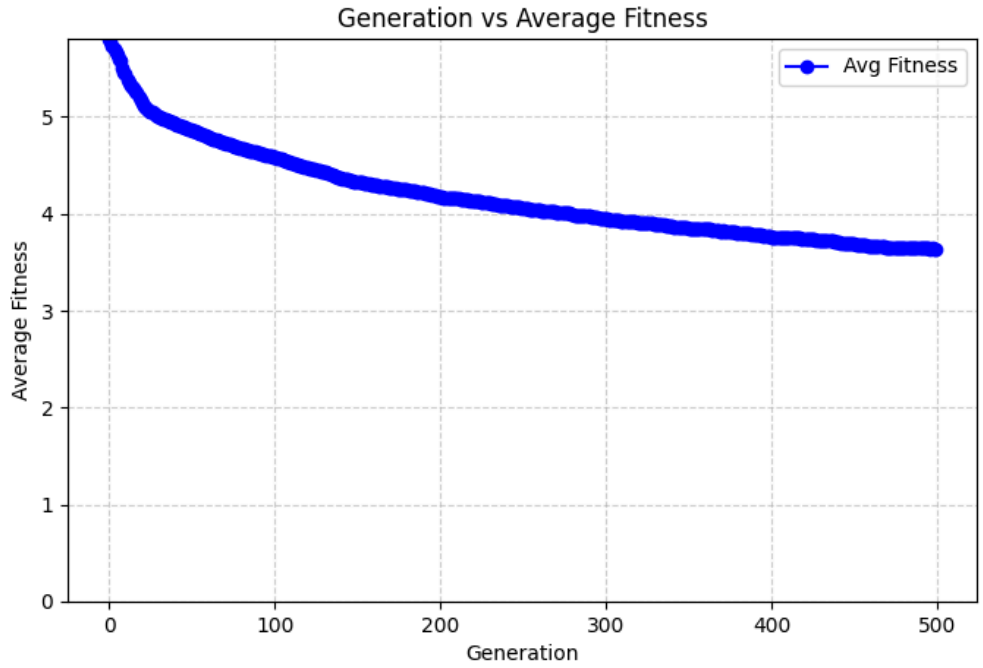
Διάγραμμα 4.2: Εξέλιξη της καταλληλότητας ανα γενιά στο GenRec1

4.3: GenRec0

Με την εκτέλεση του GenRec0 τα αποτελέσματα είναι αρκετά καλύτερα.Παρατηρείται σημαντική μείωση του MSE ανα γενία ανεξαρτήτως την επιλογή παραμέτρων.Βέβαια με την εκτέλεση μεγάλου αριθμού γενιών παρατηρείται το φαινομενο overtraining πάνω στα δεδομένα εκπαίδευσης καθώς το MSE καταλληλότητας μπορεί να μειώνεται ως και την τιμή 3(RMSE:1,7) το test score ελαχιστοποιείται στην τιμή 5,5(RMSE:2,3) και στην συνέχεια αυξάνεται πάλι όσο προχωρούν οι γενιές και το κ=μοντέλο υπερ εκπαιδεύεται σε ένα μέρος των δεδομένων.

population_size	generations	sample_ratio	similarity_penalty	elitism	lamda_reg	split_ratio	noise_scale	mutation_rate	scale	best_fitness	avg_fitness	test_score	timestamp	duration
50	100	0.6	0.1	4	0.01	0.8	0.2	0.030	0.1	4.650034	4.670636	5.499565	2025-02-19 02_49_36	00:00:56
50	100	0.8	0.4	4	0.01	0.8	0.2	0.030	0.1	4.655108	4.677918	5.502420	2025-02-19 02_46_42	00:01:12
50	100	0.6	0.4	4	0.01	0.8	0.2	0.010	0.1	4.749338	4.764147	5.716842	2025-02-19 02_55_12	00:00:54
50	100	0.6	0.4	4	0.01	0.8	0.2	0.030	0.1	4.563169	4.584721	5.775552	2025-02-19 02_45_47	00:00:54
50	100	0.6	0.4	4	0.01	0.8	0.2	0.001	0.1	5.046590	5.052491	5.802548	2025-02-19 02_54_16	00:00:55
50	100	0.4	0.4	4	0.01	0.8	0.2	0.030	0.1	4.408260	4.438919	5.829327	2025-02-19 02_48_55	00:00:40
50	100	0.6	0.4	4	0.01	0.8	0.2	0.030	0.1	4.806434	4.838750	5.831816	2025-02-19 02_51_29	00:00:55
50	100	0.6	0.4	15	0.01	0.8	0.2	0.030	0.1	4.757219	4.774799	5.871772	2025-02-19 03_02_34	00:00:55
50	100	0.6	0.4	4	0.01	0.8	0.2	0.030	0.1	4.573520	4.588839	5.877056	2025-02-19 02_53_20	00:00:55

Εικόνα 4.3.α: Αποτελέσματα GenRec0 για διαφορετικές παραμέτρους



Διάγραμμα 4.3: Εξέλιξη της καταλληλότητας ανα γενιά στο GenRec0

4.4: Σύγκριση με συνήθεις μεθόδους

Για την αξιολόγηση των αποτελεσμάτων των δύο μοντέλων στο συγκεκριμένο σύνολο δεδομένων, υλοποιήθηκαν δύο άλλες μέθοδοι που χρησιμοποιούνται στο CF για σύγκριση. Η πρώτη είναι, η μέθοδος MF Singular Value Decomposition (SVD) [24]. Η δεύτερη μέθοδος είναι μία εφαρμογή LP (Label Propagation) [12] χωρίς την χρήση γράφων και προσαρμοσμένη στις συνθήκες παλινδρόμησης του προβλήματος που επιλύεται. Ως μετρική απόστασης χρησιμοποιείται συνημιτονική ομοιότητα (cosine similarity).

Μετά την εκτέλεση τους κατέληξα στα εξής αποτελέσματα:

	Method	RMSE	MAE
0	SVD	7.566995	7.260278
1	Label Propagation	7.566995	7.260278

Εικόνα 4.4.α: Αποτελέσματα άλλων μεθόδων

Παρατηρούμε οτι τα αποτελέσματα δεν είναι ικανοποιητικά για αυτό το σύνολο δεδομένων. Ο GenRec0 εμφανίζει δραστικά χαμηλότερο RMSE της τάξης του 2 απο τις εκτελέσεις που έγινα παραπάνω, έχει βέβαια πολύ μεγαλύτερη χρονική πολυπλοκότητα σε σχέση με τις μεθόδους αυτες. Ο GenRec1 τώρα παρουσιάζει και αυτός σχετικά χαμηλότερο σφάλμα από αυτές τις μεθόδους, αλλά η εξαιρετικά μεγάλη χρονική πολυπλοκότητα και η αριθμητική αστάθεια, τον καθιστά μη χρήσιμο για οποιαδήποτε μεγάλο σύνολο δεδομένων.

Πρέπει να σημειωθεί όμως η ταχύτητα καθώς και η απλότητα των δύο μεθόδων σε σχέση με τους δύο γενετικούς αλγόριθμους καθιστώντας τες πιθανων καλύτερες επιλογές για πολύ μεγάλα σύνολα δεδομένων.

Κεφάλαιο 5: Συμπεράσματα και Πιθανές Βελτιώσεις

5.1: Συμπεράσματα

Με βάση την δημιουργία των μοντέλων καθώς και τα αποτελέσματα τους κατα την εκτέλεση μπορούμε να εξάγουμε διάφορα σημαντικά συμπεράσματα.

Καταρχάς η σύγκριση των δύο μοντέλων καταλήγει σε ξεκάθαρο “νικητή”. Το GenRec0 εμφανίζει ανώτερα αποτελέσματα σε σχέση με την ακρίβεια και είναι πολύ γρηγορότερο ως προς την χρονική πολυπλοκότητα. Αυτό το αποτέλεσμα, οφείλεται κυρίως στην δυσκολια μοντελοποίησης των σχέσεων μεταξύ όλων των χρηστών χρησιμοποιώντας πίνακες εγγύτητας τύπου W στον GenRec1. Το μοντέλο έχει απαγορευτική υπολογιστική πολυπλοκότητα και είναι αριθμητικά ασταθες λόγω των επανειλημμένων αντιστροφών σε μεγάλους αραιούς πίνακες. Αντιθέτως, ο GenRec0 μειώνει αρκετά την πολυπλοκότητα και καταλήγει σε σταθερά μαθηματικά αποτελέσματα καθιστωντας την εκπαίδευση γενετικού αλγορίθμου εφικτή και αποδοτική, μειώνοντας σταθερά το σφάλμα ανα γενιά. Η σύμπτυξη των προτιμήσεων χρηστών σε χαμηλοβαθμες μήτρες συνιστά σημαντικότατη αύξηση της απόδοσης χωρίς ουσιαστική απώλεια πληροφορίας, καθιστώντας το μοντέλο μια ενδιαφέρουσα και πρωτότυπη επιλογή για την εφαρμογή CF σε Συστήματα Συστάσεων.

5.2:Μελλοντικές βελτιώσεις και κατευθύνσεις

Κατά τον σχεδιασμό, την περάτωση και την ανάλυση των αποτελεσμάτων απο την παραπάνω διαδικασία, προέκυψαν διάφορες κατευθύνσεις και προοπτικες βελτίωσης του μοντέλου.

Καταρχάς, με την χρήση άλλων συνόλων δεδομένων, με διαφορετικά χαρακτηριστικά (απο τα δεδομένα κριτικών ταινιών του IMDb), όπου αυτά είτε θα μπορούσαν να περιέχουν κριτικές για διαφορετικά είδη αντικειμένων, είτε με διαφορετικές κλίμακες βαθμολόγησης, θα προέκυπταν συμπεράσματα για την γενικότητα της ιδέας στην οποία είναι βασισμένα τα δύο μοντέλα. Επιπλέον, η τροποποίηση της λειτουργίας τους έτσι ώστε τα μοντέλα να ενημερώνονται με νέα δεδομένα που προκύπτουν κατα την χρήση μιας εφαρμογής, δηλαδή νέα αντικείμενα κριτικές και χρήστες και στην συνέχεια, η παρατήρηση τον επιδόσεων μετά από μία τέτοια τροποποίηση θα άνοιγε το δρόμο για την πιθανή χρήση τους σε κάποια εφαρμογή κατα την χρήση της, ακόμα και για εμπορικούς σκοπούς.

Όσο αναφορά τωρα, για το μοντέλα καθαυτά, καθώς και την αλγοριθμική διαδικασία που υλοποιούν, οι πιθανές αλλαγές και βελτιώσεις είναι πολλές. Καταρχάς, στην μέθοδο που χρησιμοποιείται στον γενετικό αλγόριθμο ώς Συνάρτηση Ικανότητας, δηλαδή την διαδικασία από την οποία αξιολογείται ο κάθε πίνακας γειτνίασης W. Χρησιμοποιώντας διαφορετικές μαθηματικές διαδικασίες από αυτές που επιλέχθηκαν, θα μπορούσε η

μεταφορά πληροφορίας από τον πίνακα W, και την συσχέτιση διαφορετικών χρηστών, στην πρόβλεψη των βαθμολογιών για το κάθε αντικείμενο να εφαρμόζεται με αποδοτικότερο τρόπο, σε σχέση την ταχύτητα εκτέλεσης του μοντέλου, καθώς και όσον αφορά την εξαγωγή πληροφορίας από τον πίνακα W.Το μοντέλο GenRec 0 αποτελεί μία τέτοια βελτίωση σε σύγκριση με το GenRec1 καθώς χρησιμοποιεί μια μέθοδο MF για να παραγοντοποιήσει τον αραιό πίνακα W.Ο αλγόριθμος θα μπορούσε να επεκταθεί και για τεχνικές παραγοντοποίησης , προσαρμόζοντας τα χαρακτηριστικά του στην κάθε περίπτωση.

Στη συνέχεια, αλλαγές θα μπορούσαν και να εφαρμοστούν και στον τρόπο που λειτουργούν οι γενετικοί αλγόριθμοι που αναπτύχθηκαν. Η χρήση κάποιας διαφορετικων μεθοδων για τη διασταύρωση πινάκων γειτνίασης θα προσέδιδε σίγουρα διαφορετικά αποτελέσματα από τις υπάρχουσες. Για παράδειγμα, η χρήση ενός αλγορίθμου παρόμοιου με τον NEAT (NeuroEvolution of Augmenting Topologies)[25], θα μπορούσε να αξιοποιηθεί καλύτερα η πληροφορία που περιέχεται σε κάθε πίνακα τύπου W και να μεταφερθεί στις επόμενες γενιές πινάκων μέσω καλύτερα επιλεγμένων γονιδίων. Παρομοίως, οποιαδήποτε φάση στην λειτουργία των γενετικών αλγορίθμων θα μπορούσε να σχεδιαστεί διαφορετικά, με ποιό στοχευμένο για τη συγκεκριμένη εφαρμογή τρόπο. Η βιβλιογραφία σε σχέση με την επιλογή και βελτιστοποίηση γενετικών αλγορίθμων είναι εκτενής. Επομένως, σίγουρα οι επιλογές όσον αφορά τον σχεδιασμό των μοντέλων GenRec1 και GenRec0, μπορούν να τροποποιηθούν έτσι ώστε να εμφανίζουν βελτιωμένα αποτελέσματα.

Γενικότερα, μια άλλη προσέγγιση στο πρόβλημα της της ανάδειξης σχετικών αντικειμένων ώς προτάσεις για χρήστες θα ήταν η αντιμετώπιση του ώς πρόβλημα Κατηγοριοποίησης, και όχι σαν πρόβλημα Παλινδρόμησης όπως επέλεξα. Υπολογίζοντας δηλαδή, το δυαδικό αποτέλεσμα εάν ο κάθε χρήστης θα προτιμούσε το συγκεκριμένο αντικείμενο ή όχι, με βάση την δομή του πίνακα γειτνίασης. Αυτή η προσέγγιση, επιλέγεται πιό συχνά ούτως η άλλως σε συστήματα συστάσεων καθώς τα αποτελέσματα είναι πιο κοντά στην φύση του προβλήματος. Η ίδια προτίμηση στα προβλήματα Κατηγοριοποίησης μπορεί να παρατηρηθεί και στην εφαρμογή τεχνικών ημι-εποπτευόμενης μάθησης γενικότερα[26].

Τέλος, η σύγκριση των αλγορίθμων με παρόμοιες τεχνικές και μεθόδους που χρησιμοποιούνται στο CF, ώς προς την ποιότητα των προτάσεων αντικειμένων σε χρήστες. Αξιοποιόντας μετρικές όπως precision@n και recall@n θα προέκυπταν πιο απτά συμπεράσματα για την χρήση των μοντέλων που ανέπτυξα σε κάποιο RS.

Παρ’ όλα αυτα, στην παρούσα πτυχιακή παρουσιάζονται δύο πρωτότυπα μοντέλα **Γενετικών Αλγορίθμων** για τη δημιουργία **Πινάκων Εγγυτητας** με σκοπό την εφαρμογή **Συνεργατικού Φιλτραρίσματος** σε **Συστήματα Συστάσεων**.

Πίνακας Ορολογίας

Ξενόγλωσσος Όρος	Ελληνικός Όρος
Recommender/Recommendation Systems	Συστήματα Συστάσεων
Machine Learning	Μηχανική Μάθηση
Supervised Learning	Εποπτευόμενη Μάθηση
Unsupervised Learning	Μη Εποπτευόμενη Μάθηση
Semi-supervised Learning	Ημι-εποπτευόμενη Μάθηση
Fitness Function	Συνάρτηση Ικανότητας
Affinity Matrix	Μήτρα Εγγύτητας
Collaborative Filtering	Συνεργατικό Φιλτράρισμα
Genetic Algorithms	Γενετικοί Αλγόριθμοι
Population	Πληθυσμός
Individuals	Άτομα
Generation	Γενιά
Fitness Function	Συνάρτηση Καταλληλότητας
Fitness value	Τιμή Καταλληλότητας
Low-Rank Embeddings	Χαμηλόβαθμες Υπερπαραμέτροι Πινάκων
Chromosome	Χρωμόσωμα
Mutation Rate	Βαθμός Μετάλλαξης
Label Propagation	Διάδοση Ετικετών
Matrix Factorization	Παραγοντοποίηση Πινάκων

Πίνακας αρκτικόλεξων-συντμήσεων-ακρωνυμίων

RS	Recommender Systems
ML	Machine Learning
CF	Collaborative Filtering
GA	Genetic Algorithms
AI	Artificial Intelligence
SL	Supervised Learning
UL	Unsupervised Learning
SSL	Semi-Supervised Learning
LP	Label Propagation
MF	Matrix Factorization
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error

Βιβλιογραφία:

[1] D. Jannach, P. Pu, F. Ricci, and M. Zanker, “Recommender systems: Past, present, future,” *AI Mag.*, vol. 42, no. 3, pp. 3–6, 2021, doi: 10.1609/aimag.v42i3.18139.

[2] R. and Markets, “AI-based Recommendation System Global Research Report 2024: A $3.28 Billion Market in 2028 - Long-term Forecast to 2033 with Alphabet, Microsoft, Alibaba, Meta, AWS Leading,” GlobeNewswire News Room. Accessed: Jan. 12, 2025. [Online]. Available: https://www.globenewswire.com/news-release/2024/06/27/2905080/28124/en/AI-based-Recommendation-System-Global-Research-Report-2024-A-3-28-Billion-Market-in-2028-Long-term-Forecast-to-2033-with-Alphabet-Microsoft-Alibaba-Meta-AWS-Leading.html

[3] Y. Li, K. Liu, R. Satapathy, S. Wang, and E. Cambria, “Recent Developments in Recommender Systems: A Survey,” Jun. 22, 2023, *arXiv*: arXiv:2306.12680. doi: 10.48550/arXiv.2306.12680.

[4] Z. Dong, Z. Wang, J. Xu, R. Tang, and J. Wen, “A Brief History of Recommender Systems,” Sep. 05, 2022, *arXiv*: arXiv:2209.01860. doi: 10.48550/arXiv.2209.01860.

[5] “(PDF) Evaluating Collaborative FilteringRecommender Algorithms: A Survey,” *ResearchGate*, Oct. 2024, doi: 10.1109/ACCESS.2018.2883742.

[6] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li, “A survey of matrix completion methods for recommendation systems,” *Big Data Min. Anal.*, vol. 1, no. 4, pp. 308–323, Dec. 2018, doi: 10.26599/BDMA.2018.9020008.

[7] G. Suganeshwari and S. P. Syed Ibrahim, “A Survey on Collaborative Filtering Based Recommendation System,” in *Proceedings of the 3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC – 16’)*, vol. 49, V. Vijayakumar and V. Neelanarayanan, Eds., in Smart Innovation, Systems and Technologies, vol. 49. , Cham: Springer International Publishing, 2016, pp. 503–518. doi: 10.1007/978-3-319-30348-2_42.

[8] S. Karagiannakos, “An introduction to Recommendation Systems: an overview of machine and deep learning architectures,” AI Summer. Accessed: Nov. 21, 2024. [Online]. Available: https://theaisummer.com/recommendation-systems/

[9] “Definition and History of Recommender Systems,” Computer Science. Accessed: Nov. 30, 2024. [Online]. Available: https://international.binus.ac.id/computer-science/2020/11/03/definition-and-history-of-recommender-systems/

[10] N. Quoc, P. Hoai, and H. Xuan, “Statistical Implicative Similarity Measures for User-based Collaborative Filtering Recommender System,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 11, 2016, doi: 10.14569/IJACSA.2016.071118.

[11] S. Juumta and D. Faggella, “What is Machine Learning? - An Informed Definition,” Emerj Artificial Intelligence Research. Accessed: Jan. 23, 2025. [Online]. Available: https://emerj.com/what-is-machine-learning/

[12] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, “Learning with Local and Global Consistency”.

[13] Y. Fujiwara and F. Yasuhiro, “Efficient Label Propagation”.

[14] O. Chapelle, A. Zien, and B. Schölkopf, Eds., *Semi-supervised learning*. in Adaptive computation and machine learning series. Cambridge, Massachusetts: MIT Press, 2006.
[15] Y. Zhang, “An Introduction to Matrix factorization and Factorization Machines in Recommendation System, and Beyond,” Mar. 12, 2022, *arXiv*: arXiv:2203.11026. doi: 10.48550/arXiv.2203.11026.
[16] L. Haldurai, T. Madhubala, and R. Rajalakshmi, “A Study on Genetic Algorithm and its Applications,” vol. 4, 2016.
[17] A. Dahiya and S. Sangwan, “Literature Review on Genetic Algorithm,” vol. 05, no. 16, 2018.
[18] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “DeepFM: A Factorization-Machine based Neural Network for CTR Prediction,” in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, Aug. 2017, pp. 1725–1731. doi: 10.24963/ijcai.2017/239.
[19] B. Alhijawi and Y. Kilani, “Using genetic algorithms for measuring the similarity values between users in collaborative filtering recommender systems,” in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Okayama, Japan: IEEE, Jun. 2016, pp. 1–6. doi: 10.1109/ICIS.2016.7550751.
[20] B. J. M. Alhijawi, “The Use of the Genetic Algorithms in the Recommender Systems”.
[21] D. Tang, B. Qin, T. Liu, and Y. Yang, “User Modeling with Neural Network for Review Rating Prediction”.
[22] F. Rothlauf, Ed., *Representations for Genetic and Evolutionary Algorithms*. in SpringerLink Bücher. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2006. doi: 10.1007/3-540-32444-5.
[23] Walchand College of Engineering, U. A.J., S. P.D., and Government College of Engineering, Karad, “CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW,” *ICTACT J. Soft Comput.*, vol. 06, no. 01, pp. 1083–1092, Oct. 2015, doi: 10.21917/ijsc.2015.0150.
[24] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Application of Dimensionality Reduction in Recommender System - A Case Study:,” Defense Technical Information Center, Fort Belvoir, VA, Jul. 2000. doi: 10.21236/ADA439541.
[25] K. O. Stanley and R. Miikkulainen, “Evolving Neural Networks through Augmenting Topologies,” *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002, doi: 10.1162/106365602320169811.
[26] L. Wasserman and J. Lafferty, “Statistical Analysis of Semi-Supervised Regression,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2007. Accessed: Jan. 23, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2007/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html

Παραρτήματα

Παράρτημα Α: Πηγαίος Κώδικας σε Python

https://github.com/StavrosTZI/GenRec