ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Πανεπιστήμιο Πειραιώς

Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών «Πληροφορικής»

ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

Ονοματεπώνυμο : Μαρία Αμοργιανού

Αριθμός Μητρώου : ΜΠΠΛ2205

Ονοματεπώνυμο : Αναστάσιος Αντωνίου

Αριθμός Μητρώου : ΜΠΠΛ2206

Ονοματεπώνυμο : Αντώνιος Τζιβάκης

Αριθμός Μητρώου : ΜΠΠΛ2244

Προ-επεξεργασία Δεδομένων

1. Να βρείτε το σύνολο των μοναδικών χρηστών και το σύνολο των μοναδικών αντικειμένων .

Κλάσεις και Μέθοδοι που Χρησιμοποιούνται

1. ImportService:

Αυτή η κλάση διαχειρίζεται την εισαγωγή των δεδομένων από τα αρχεία. Χρησιμοποιεί μεθόδους όπως importFromFolder1 και importFromFolder2 για να διαβάσει τα δεδομένα των ταινιών και των αξιολογήσεων από τα αρχεία.

```
def importFromFolder2(self):
    print("Beginning to read the reviews from folder 2")
    filereader = FileReader("2_reviews_per_movie_raw")

    #key filename,rows list
    rowsDict = filereader.retrieveData()
    self.parseReviewsDictToDTOS(rowsDict)
    #print(len(self._reviewsPerMovieDTOS))
    self.parseReviewsDTOsToMoviesRepository()
    self.parseReviewsDTOsToReviewsRepository()
```

Στο παραπάνω τμήμα, η μέθοδος importFromFolder2 διαβάζει τις αξιολογήσεις ταινιών από τον φάκελο 2_reviews_per_movie_raw. Τα δεδομένα αυτά μετατρέπονται σε αντικείμενα ReviewsPerMovieDTO και στη συνέχεια προστίθενται στις κατάλληλες αποθήκες (Repositories) για χρήστες και ταινίες.

2. MoviesRepository,ReviewsRepository, UsersRepository:

Οι αποθήκες (repositories) αυτές χρησιμοποιούνται για να αποθηκεύουν και να διαχειρίζονται τις πληροφορίες των ταινιών, των αξιολογήσεων και των χρηστών αντίστοιχα. Κάθε φορά που προστίθεται μια νέα ταινία ή μια νέα αξιολόγηση, τα δεδομένα αποθηκεύονται στις αντίστοιχες αποθήκες.Τα mappers είναι οι

ενδιάμεσες κλάσεις που χρησιμεύουν για την αντιστοίχιση των entries στα data με τα αντίστοιχα models που αποθηκεύονται μετά με την addX(e.g. addMovie) στα repositories.

```
def addMovie(self,movie=Movie):
    if not self._movies:
        movie.Id=1
    else:
        movie.Id=self._movies[-1].Id+1
        #print(f"Added movie with id={movie.Id}")
    self._movies.append(movie)
    self.writeMovieToDicts(movie)
```

Η παραπάνω μέθοδος addMovie στην κλάση MoviesRepository προσθέτει μια νέα ταινία στο σύστημα. Αντίστοιχες μέθοδοι υπάρχουν και στις άλλες αποθήκες για χρήστες και αξιολογήσεις.

3. UsersService:

Η κλάση αυτή αναλαμβάνει να "χαρτογραφήσει" τις αξιολογήσεις που υπάρχουν στην αποθήκη αξιολογήσεων (ReviewsRepository) στους αντίστοιχους χρήστες στην αποθήκη χρηστών (UsersRepository).

```
def writeReviewDataToUsers(self):
    for review in self._reviewsRepository.Reviews:
        self.mapReviewToUser(review)
    self._context.usersRepository=self._usersRepository
```

Η μέθοδος writeReviewDataToUsers χρησιμοποιείται για να διασφαλίσει ότι κάθε χρήστης έχει καταχωρηθεί με τις αντίστοιχες αξιολογήσεις του, ώστε να μπορούμε να έχουμε το πλήρες σύνολο των χρηστών U με τις αξιολογήσεις τους.

Βήματα για την Εύρεση των Συνόλων

1. **Εισαγωγή Δεδομένων**:

Ο κώδικας εισάγει τα δεδομένα αξιολογήσεων και ταινιών από τα αρχεία. Οι χρήστες και οι ταινίες καταχωρούνται στις αποθήκες τους με μοναδικά IDs.

2. **Χρήση των Αποθηκών**:

Μετά την εισαγωγή των δεδομένων, τα αντικείμενα *UsersRepository* και *MoviesRepository* περιέχουν το σύνολο των μοναδικών χρηστών U και των μοναδικών αντικειμένων I αντίστοιχα.

3. **Αξιοποίηση των Δεδομένων**:

Για να πάρουμε το πλήθος των μοναδικών χρηστών και των ταινιών, να χρησιμοποιούμε τις μεθόδους που επιστρέφουν το σύνολο των αντικειμένων στις αποθήκες (self._usersRepository.Users και *self._moviesRepository.Movies*).

2. Θεωρείστε την συνάρτηση Φ:U→P(I) (όπου P(I) το δυναμοσύνολο του I) η οποία ∀ u∈U επιστρέφει το σύνολο φ(u) ⊂ I των αντικειμένων που αξιολογήθηκαν από τον χρήστη u. Μπορούμε να γράψουμε, επομένως, ότι φ(u) = {i ∈ I: R(u,i) > 0}. Να περιορίσετε τα σύνολα των μοναδικών χρηστών U και μοναδικών αντικειμένων I στα αντίστοιχα σύνολα Uˆ και Iˆ έτσι ώστε ∀ u ∈ Uˆ, R_min ≤ |φ(u)| ≤ R_max όπου R_min και R_max ο ελάχιστος απαιτούμενος και ο μέγιστος επιτρεπτός αριθμός αξιολογήσεων ανά χρήστη. Θεωρήστε προφανώς ότι |Uˆ| = n < N και |Iˆ| = m < M.

Τμήματα του Κώδικα που Χρησιμοποιούνται

1. Μέθοδος *writeReviewDataToUsers* στην *UsersService*:

- Αυτή η μέθοδος συνδέει τις αξιολογήσεις που έχουν αποθηκευτεί στο *ReviewsRepository* με τους αντίστοιχους χρήστες στο *UsersRepository.*
- Κατά την αντιστοίχιση κάθε αξιολόγησης σε έναν χρήστη, δημιουργείται ουσιαστικά το σύνολο φ(u) για κάθε χρήστη u.

2. **Μέθοδος *mapReviewToUser* στην *UsersService*:**
   - Αυτή η μέθοδος είναι υπεύθυνη για την αντιστοίχιση μιας αξιολόγησης σε έναν χρήστη. Εάν ο χρήστης έχει αξιολογήσει την ταινία, αυτή η αξιολόγηση προστίθεται στο αντίστοιχο σύνολο φ(u).

```python
def mapReviewToUser(self,review:Review):
    user=self._usersRepository.findUserByUsername(review.username)
    moviesLength = len(self._moviesRepository.Movies)
    movieTitle = review.movieTitle
    movie = self._moviesRepository.findMovieByTitle(movieTitle)



    if user is None:
        user=User(moviesLength)
        user.Username=review.username
    if movie is not None:
        index = movie.Id-1
        ratings = user.Ratings
        ratings[index]=int(review.rating)
        timestamps = user.TimeStamps
        # timestamps[index] = datetime.strptime(review.date, '%d %B %Y')
        timestamps[index] = review.date
        user.TimeStamps=timestamps
        user.Ratings=ratings
    else:
        print("movie is none with title:",movieTitle)
    # print(user.Ratings)
    user.reviews.append(review)
    user.total_reviews=len(user.reviews)
    self._usersRepository.save(user)
```

Το σύνολο ϕ(u) δημιουργείται ως ένα διάνυσμα αξιολογήσεων (ratings) για τον χρήστη u, όπου το ϕ(u) περιλαμβάνει μόνο τα αντικείμενα (ταινίες) που έχουν βαθμολογηθεί με τιμή μεγαλύτερη από 0.

3. **Μέθοδος *filterData* στην *UsersService*:**

- Η μέθοδος αυτή χρησιμοποιείται για να περιορίσει το σύνολο των χρηστών U στο U^, κρατώντας μόνο εκείνους τους χρήστες των οποίων ο αριθμός των αξιολογήσεων ανήκει στο διάστημα [Rmin,Rmax].

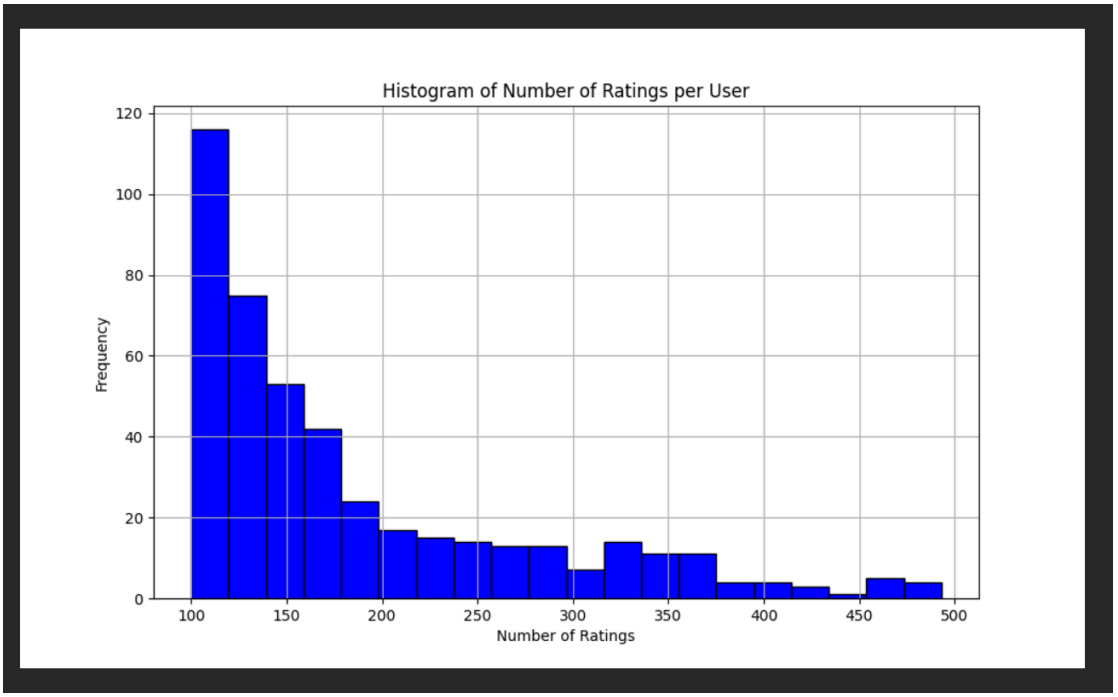
```
def filterData(self,min,max):
    users = self._usersRepository.Users
    filteredUsers=[]
    for user in users:
        if min<=np.count_nonzero(user.Ratings)<=max:
            filteredUsers.append(user)
    self._usersRepository.Users=filteredUsers
```

Η *filterData* ελέγχει τον αριθμό των αξιολογήσεων που έχει κάθε χρήστης (δηλαδή, το μέγεθος του συνόλου ϕ(u)) και κρατά μόνο τους χρήστες που πληρούν τα κριτήρια Rmin ≤ |ϕ(u)| ≤ Rmax.

3. Να δημιουργήσετε και να αναπαραστήσετε γραφικά τα ιστογράμματα συχνοτήτων για το πλήθος αλλά και για το χρονικό εύρος των αξιολογήσεων του κάθε χρήστη.

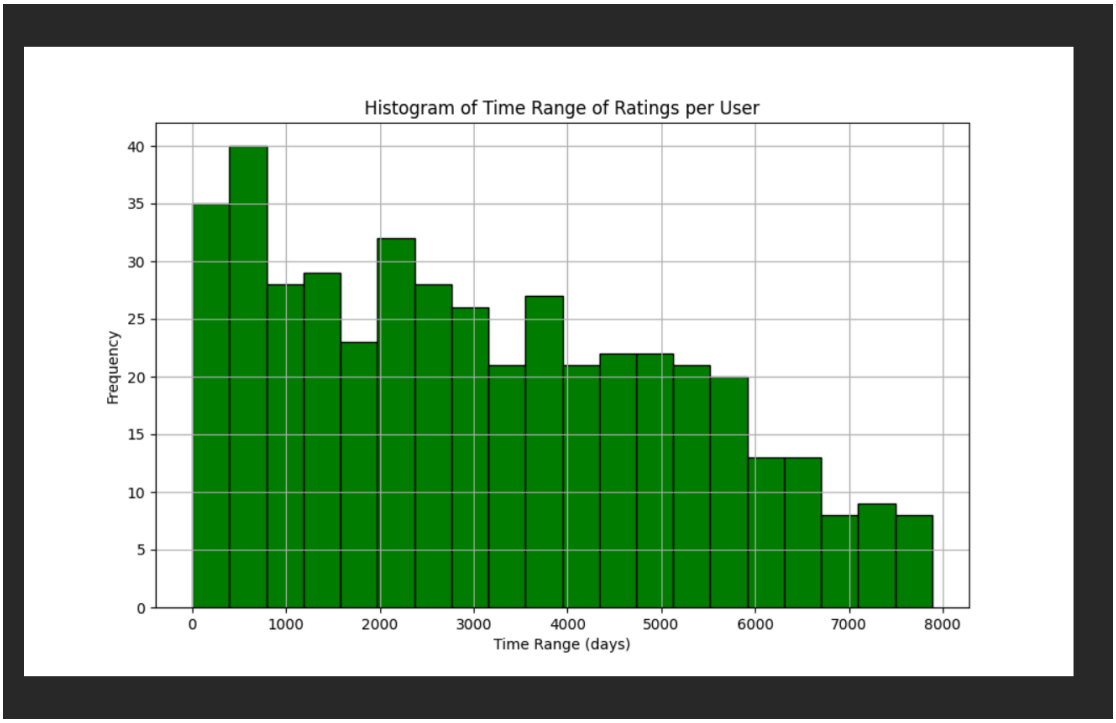
Για το παραπάνω ερώτημα τρέχοντας τον κώδικα κι έχοντας θέσει ως *Rmin* τις 100 ψήφους ανά χρήστη ενώ ως *Rmax* τις 500 τα ιστογράμματα που λάβαμε είναι τα παρακάτω:

1. Histogram of Number of Ratings per User (Πλήθος Αξιολογήσεων ανά Χρήστη):



Αυτό το γράφημα δείχνει τον αριθμό αξιολογήσεων που έχουν υποβληθεί από κάθε χρήστη. Παρατηρούμε ότι η πλειοψηφία των χρηστών έχει υποβάλει γύρω στις 100-150 αξιολογήσεις, και καθώς ο αριθμός των αξιολογήσεων αυξάνεται, ο αριθμός των χρηστών μειώνεται. Αυτή η κατανομή είναι συνηθισμένη σε συστήματα αξιολόγησης, όπου ένα μικρό ποσοστό των χρηστών είναι υπερ-δραστήριοι και υποβάλλουν πολλές αξιολογήσεις, ενώ η πλειοψηφία των χρηστών υποβάλλει μόνο έναν μικρό αριθμό αξιολογήσεων.

2. Histogram of Time Range of Ratings per User (Χρονικό Εύρος Αξιολογήσεων ανά Χρήστη):



Αυτό το γράφημα δείχνει το εύρος του χρόνου (σε ημέρες) κατά τον οποίο οι χρήστες παρείχαν τις αξιολογήσεις τους. Παρατηρούμε ότι οι περισσότεροι χρήστες έχουν ένα σχετικά μικρό χρονικό εύρος αξιολογήσεων, γύρω στις 0-2000 ημέρες. Καθώς το εύρος αυξάνεται, ο αριθμός των χρηστών που συνεχίζουν να αξιολογούν μειώνεται. Αυτή η κατανομή δείχνει ότι οι περισσότεροι χρήστες είναι πιο ενεργοί για μικρότερες χρονικές περιόδους, ενώ λίγοι χρήστες παραμένουν ενεργοί για μεγαλύτερες περιόδους.

4. Δημιουργήστε μια εναλλακτική αναπαράσταση του συνόλου των δεδομένων ως ένα σύνολο διανυσμάτων προτιμήσεων με $\boldsymbol{Rj=R}(uj)\in\boldsymbol{Ro}$ *,*$\forall j\in[n]$*. Συγκεκριμένα, μπορούμε να γράψουμε ότι: Rj(k)={R(uj,ik) αν R(uj,ik)>0, 0 αν R(uj,ik)=0}*

Μέθοδος *exractRatings* στην *UsersService*:

Αυτή η μέθοδος είναι υπεύθυνη για την εξαγωγή των αξιολογήσεων από τους χρήστες και τη δημιουργία ενός πίνακα διανυσμάτων R. Κάθε γραμμή του πίνακα R αντιστοιχεί σε ένα διάνυσμα προτιμήσεων Rj για τον χρήστη uj.

```python
def exractRatings(self):
    # ratings = np.zeros(len(self._usersRepository.Users),len(self._usersRepository.Users[0]))
    users = len(self._usersRepository.Users)
    movies = len(self._moviesRepository.Movies)
    ratings = np.zeros((users,movies),dtype=int)

    for index,user in enumerate(self._usersRepository.Users):
        ratings[index, :]=user.Ratings
```

Σε αυτό το τμήμα του κώδικα, η exractRatings δημιουργεί έναν πίνακα R με διαστάσεις n × m, όπου n είναι ο αριθμός των χρηστών και m ο αριθμός των ταινιών.

Κάθε γραμμή του πίνακα αντιπροσωπεύει τις αξιολογήσεις ενός χρήστη για όλες τις ταινίες. Αν μια ταινία δεν έχει αξιολογηθεί από τον χρήστη, τότε η αντίστοιχη τιμή στο διάνυσμα είναι 0.

Αλγόριθμοι Ομαδοποίησης Δεδομένων

c. Να αναπαραστήσετε γραφικά τις συστάδες των χρηστών που αναγνωρίστηκαν από τον αλγόριθμο k-means για κάθε μία από τις παραπάνω μετρικές για διάφορες τιμές της παραμέτρου L.

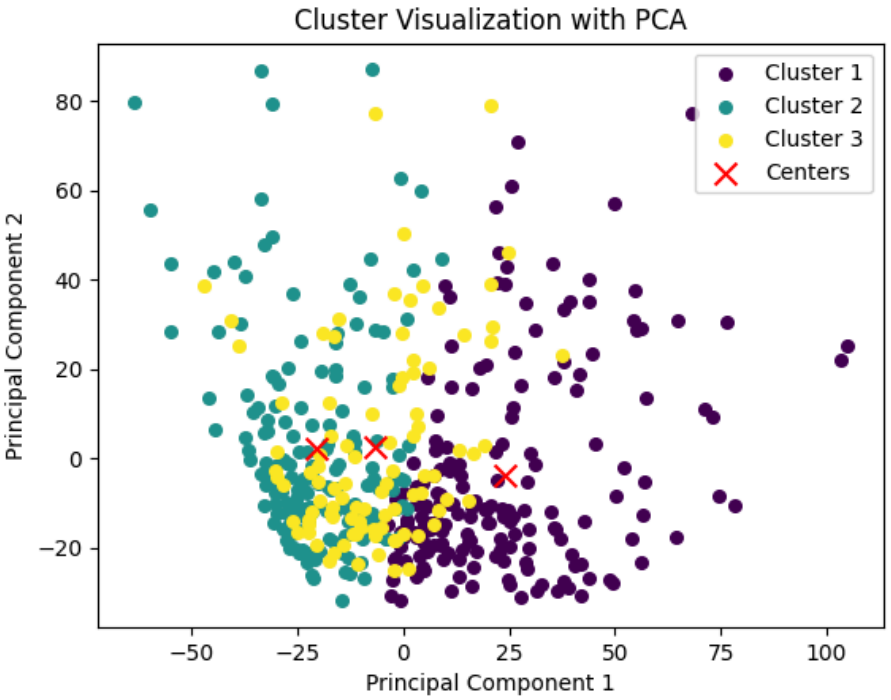
Για να οργανώσουμε το περιορισμένο πλήθος των χρηστών σε L συστάδες στην κλάση *ClusterService*, χρησιμοποιούμε τη μέθοδο *applyKmeans* για να ομαδοποιήσει τους χρήστες βασισμένους στις αξιολογήσεις τους. Η μέθοδος αυτή λαμβάνει ως είσοδο τον αριθμό των συστάδων L και χρησιμοποιεί την αναπαράσταση R για την ομαδοποίηση.

```python
def applyKmeans(self,numOfClusters):
    self._R = np.nan_to_num(self._R)  # Replace NaNs with 0
    R=self._R


    # Create instance of distance metric with your custom function
    metric = distance_metric(type_metric.USER_DEFINED, func=self._metric)

    # Prepare initial centers using K-Means++ method
    # initial_centers = kmeans_plusplus_initializer(R, numOfClusters).initialize()
    initial_centers = self.kmeans_plus_plus(R,numOfClusters)

    # Create instance of K-Means algorithm with prepared centers
    kmeans_instance = kmeans(R, initial_centers, metric=metric)

    # Run cluster analysis and obtain results
    kmeans_instance.process()
    clusters = kmeans_instance.get_clusters()
    final_centers = kmeans_instance.get_centers()
    # self.checkCenters(initial_centers,R,numOfClusters)
    self._clusters=clusters
    self._centers=final_centers

    # Display results
    self._logger.appendToFile("clusters.txt")
    self._logger.writeLine(("Number of Clusters:"+ str(len(clusters))))
    self._logger.writeObject("clusters:",final_centers,4)
    self._logger.close()
```

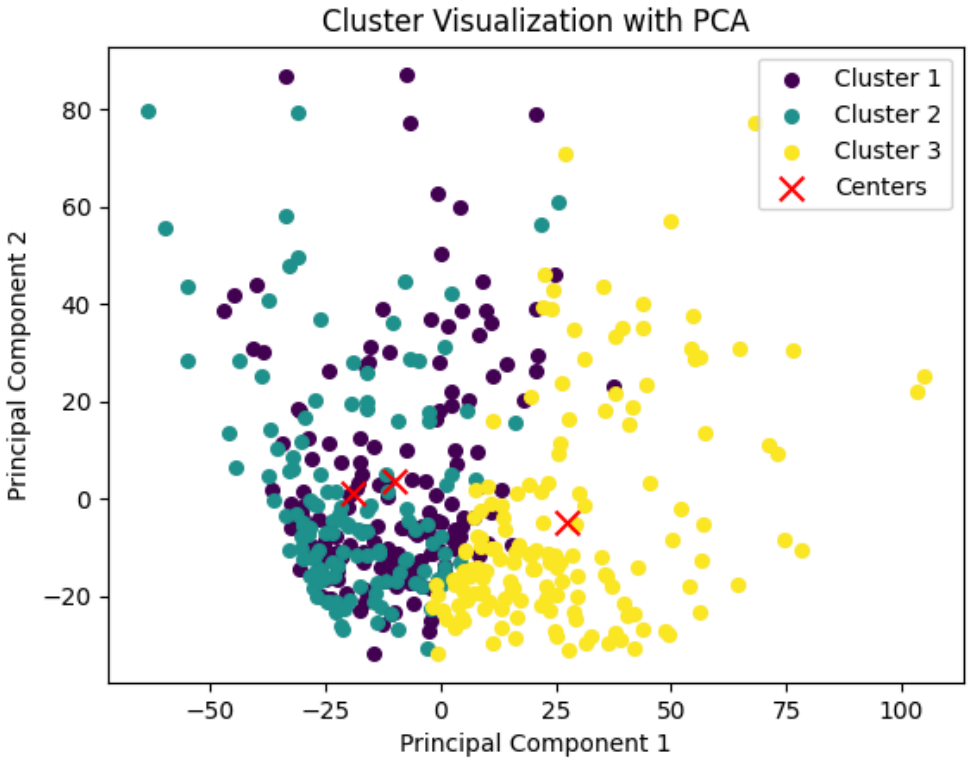
Η μέθοδος αυτή δημιουργεί συστάδες χρηστών, εφαρμόζοντας τον αλγόριθμο K-Means πάνω στα διανύσματα προτιμήσεων των χρηστών, όπως αυτά αναπαρίστανται στο R.

Στη συνέχεια έχοντας επιλέξει τη δημιουργία 3 συστάδων οι γραφικές αναπαραστάσεις που προκύπτουν βάσει των μετρικών Euclidean Distance και Cosine Distance τρέχοντας τον κώδικα είναι:

1. Graph using Euclidean Distance



2. Graph using Cosine Distance



d. Να σχολιάσετε την αποτελεσματικότητα των συγκεκριμένων μετρικών στην αποτίμηση της ομοιότητας μεταξύ ενός ζεύγους διανυσμάτων προτιμήσεων χρηστών Ru και Rv.

Γράφημα 1(Μετρική Ευκλείδειας Απόστασης):

- **Κατανομή Συστάδων:**

Οι συστάδες είναι πιο ξεκάθαρα διαχωρισμένες σε σύγκριση με το πρώτο γράφημα. Η Συστάδα 3 (κίτρινο) είναι πιο διασκορπισμένη, αλλά εξακολουθεί να είναι κατά κύριο λόγο διαχωρισμένη από τις άλλες συστάδες. Η Συστάδα 1 (μωβ) και η Συστάδα 2 (γαλαζοπράσινο) έχουν λιγότερη επικάλυψη, δείχνοντας ότι η Ευκλείδεια απόσταση αποτυπώνει καλύτερα τις διαφορές μεταξύ

αυτών των ομάδων στον χώρο χαρακτηριστικών. Αυτό υποδηλώνει ότι η Ευκλείδεια απόσταση μπορεί να είναι πιο αποτελεσματική στον διαχωρισμό αυτών των συγκεκριμένων σημείων δεδομένων, επειδή λαμβάνει υπόψη τις απόλυτες διαφορές στις τιμές των χαρακτηριστικών (βαθμολογίες).

- **Κέντρα Συστάδων:**

Τα κέντρα είναι πιο διαχωρισμένα σε σχέση με το πρώτο γράφημα, υποδεικνύοντας ότι η μετρική Ευκλείδειας απόστασης παρέχει μια πιο σαφή διάκριση μεταξύ των διαφορετικών ομάδων στον χώρο που μειώνεται από την ανάλυση PCA.

Κύριες Διαφορές:

- **Διαχωρισμός:**

Η μετρική Ευκλείδειας απόστασης (Γράφημα 2) οδηγεί σε καλύτερο διαχωρισμό μεταξύ των συστάδων στον χώρο PCA σε σύγκριση με τη μετρική Συνημιτόνου (Γράφημα 1). Αυτό μπορεί να υποδηλώνει ότι η Ευκλείδεια απόσταση είναι πιο κατάλληλη για το σύνολο δεδομένων και τον χώρο χαρακτηριστικών σας, τουλάχιστον στο πλαίσιο της ανάλυσης PCA.

- **Επικάλυψη:**

Υπάρχει περισσότερη επικάλυψη στις συστάδες με τη μετρική Συνημιτόνου, κάτι που μπορεί να υποδηλώνει ότι οι συστάδες δεν είναι τόσο διακριτές ή ότι οι διαφορές μεταξύ των ομάδων είναι περισσότερο γωνιακές (κατευθυντικές) παρά βασισμένες στο μέγεθος.

- **Διασπορά Συστάδων:**

Οι συστάδες στο γράφημα της Ευκλείδειας απόστασης φαίνονται να έχουν πιο ομοιόμορφη κατανομή, ενώ στο γράφημα της Συνημιτόνου μία από τις συστάδες (πιθανόν η κίτρινη) φαίνεται να κυριαρχεί και να εξαπλώνεται σε μεγαλύτερη περιοχή.

Γράφημα 2 (Μετρική Συνημιτόνου):

- **Κατανομή Συστάδων:**

Υπάρχει σημαντική επικάλυψη μεταξύ των συστάδων, ειδικά μεταξύ της Συστάδας 1 (μωβ) και της Συστάδας 3 (κίτρινο), καθώς και κάποια επικάλυψη μεταξύ της Συστάδας 2 (γαλαζοπράσινο) και των άλλων. Αυτό υποδηλώνει ότι όταν χρησιμοποιείται η μετρική συνημιτόνου, ο αλγόριθμος συσταδοποίησης δυσκολεύεται να διακρίνει διαφορετικές ομάδες βάσει των χαρακτηριστικών που παρέχονται.

Οι συστάδες είναι πιο αναμεμειγμένες, κάτι που υποδεικνύει ότι τα σημεία δεδομένων δεν διαχωρίζονται τόσο καλά στο χώρο που προκύπτει από την ανάλυση PCA. Αυτό μπορεί να οφείλεται στο γεγονός ότι η ομοιότητα συνημιτόνου μετρά τη γωνία μεταξύ των διανυσμάτων παρά την απόσταση, γεγονός που μπορεί να οδηγήσει σε λιγότερο διακριτό διαχωρισμό όταν το μέγεθος των διανυσμάτων (βαθμολογίες) δεν είναι τόσο σημαντικό.

- **Κέντρα Συστάδων:**

Τα κέντρα (κόκκινοι σταυροί) είναι πιο κοντά το ένα στο άλλο, κάτι που μπορεί να υποδεικνύει ότι ο υπολογισμός του κέντρου στο χώρο PCA δεν παράγει καλά διαχωρισμένα κέντρα. Αυτό μπορεί να συμβάλλει στις επικαλυπτόμενες συστάδες.

Συμπέρασμα:

- Η Ευκλείδεια Απόσταση μπορεί να παρέχει πιο καθορισμένες συστάδες σε αυτό το σενάριο, οδηγώντας σε πιο σαφείς διακρίσεις μεταξύ των σημείων δεδομένων.
- Η Συνημιτόνου Απόσταση, από την άλλη, μπορεί να είναι πιο χρήσιμη αν ενδιαφέρεστε για την κατανόηση της κατευθυντικής ομοιότητας μεταξύ των σημείων δεδομένων, αλλά μπορεί να οδηγήσει σε λιγότερο σαφείς συστάδες.

Αλγόριθμοι Παραγωγής Συστάσεων με Χρήση Τεχνητών Νευρωνικών Δικτύων

2. Να δημιουργήσετε μια εναλλακτική οργάνωση του περιορισμένου συνόλου των χρηστών σε L συστάδες $U=G1∪G2∪⋯∪GL$ έτσι ώστε $Ga∩Gb≠∅,∀a,∈[]$ με $a ≠ b$ κάνοντας χρήση της παρακάτω μετρικής (**): $dist(u,v)=1−|φ(u)∩φ(v)|/|φ(u)∪φ(v)|$ (5)

Στον κώδικα η μετρική Jaccard υλοποιείται στο ***JaccardDistance.py*** με τη χρήση της κλάσης ***JaccardSimilar***:

```python
class JaccardSimilar:

    def __init__(self):
        self._R=None
        self._binary_R=None
```

```python
def calculate_jaccard_distance(self):
    R=self._R
    # Convert ratings matrix to binary (1 if rated, 0 otherwise)
    binary_ratings = (R > 0).astype(int)
    self._binary_R=binary_ratings
    # Initialize a matrix to store distances
    num_users = R.shape[0] #υπολογίζουμε το σύνολο των χρηστών
    distance_matrix = np.zeros((num_users, num_users))

    for i in range(num_users):
        for j in range(i + 1, num_users):
            # Calculate intersection and union for users i and j
            intersection = np.sum(np.logical_and(binary_ratings[i], binary_ratings[j])) #τομή συνόλου στη σχέση 5
            union = np.sum(np.logical_or(binary_ratings[i], binary_ratings[j])) #ένωση συνόλου στη σχέση 5

            # Calculate Jaccard index and then distance
            if union == 0:
                # Avoid division by zero; implies both users have rated no items
                jaccard_index = 1
            else:
                jaccard_index = intersection / union

            # Jaccard distance
            distance = 1 - jaccard_index

            # Fill symmetric distance matrix
            distance_matrix[i, j] = distance_matrix[j, i] = distance

    return distance_matrix
```

Αυτός ο κώδικας:

- Μετατρέπει τις αξιολογήσεις σε δυαδική μορφή (1 αν ο χρήστης αξιολόγησε την ταινία, 0 αν όχι).
- Υπολογίζει την απόσταση Jaccard μεταξύ κάθε ζεύγους χρηστών.
- Επιστρέφει τον τετραγωνικό πίνακα αποστάσεων.

a) Να εξηγήσετε τι εκφράζει η συγκεκριμένη μετρική και να προσδιορίσετε τα μειονεκτήματά της σε σχέση με τις μετρικές που περιγράφονται στις σχέσεις (2) και (4) υπό το πρίσμα της ιδιαίτερης οργάνωσης των χρηστών που μπορεί να επιφέρει.

Η μετρική Jaccard $dist(u,v) = 1 - |\phi(u) \cap \phi(v)| / |\phi(u) \cup \phi(v)|$ εκφράζει την ομοιότητα ή διαφορά μεταξύ δύο συνόλων δεδομένων, στην περίπτωσή μας τις ταινίες που έχουν αξιολογηθεί από δύο χρήστες, uuu και vvv.

Ερμηνεία:

- Το $\phi(u)$ είναι το σύνολο των ταινιών που έχει αξιολογήσει ο χρήστης uuu.
- Το $\phi(v)$ είναι το σύνολο των ταινιών που έχει αξιολογήσει ο χρήστης vvv.
- Η τομή $|\phi(u) \cap \phi(v)|$ είναι το πλήθος των ταινιών που έχουν αξιολογηθεί και από τους δύο χρήστες.
- Η ένωση $|\phi(u) \cup \phi(v)|$ είναι το πλήθος των ταινιών που έχουν αξιολογηθεί από τουλάχιστον έναν από τους δύο χρήστες.
- Η μετρική Jaccard, λοιπόν, μετρά το ποσοστό της κοινής αξιολόγησης ως προς το συνολικό πλήθος των μοναδικών αξιολογήσεων. Όσο πιο κοντά είναι στο 0, τόσο πιο όμοιοι είναι οι δύο χρήστες.

Πλεονεκτήματα της Μετρικής Jaccard

1. **Αντιμετώπιση της Αραιότητας**: Η Jaccard είναι ιδιαίτερα χρήσιμη σε αραιά δεδομένα όπου οι χρήστες έχουν αξιολογήσει μόνο ένα μικρό υποσύνολο των ταινιών. Σε τέτοια δεδομένα, άλλες μετρικές μπορεί να υπερεκτιμούν τις διαφορές, ενώ η Jaccard εστιάζει μόνο στα κοινά στοιχεία.
2. **Δυαδική Ανάλυση**: Επικεντρώνεται στο αν οι χρήστες έχουν αξιολογήσει παρόμοιες ταινίες ανεξάρτητα από τις συγκεκριμένες τιμές των αξιολογήσεων. Αυτό είναι χρήσιμο όταν οι αξιολογήσεις είναι δυαδικές ή όταν μας ενδιαφέρει μόνο το αν ένας χρήστης έχει δει μία ταινία.

Μειονεκτήματα της Μετρικής Jaccard σε Σχέση με τις Μετρικές Ευκλείδεια και Συνημίτονο

1. **Αγνόηση της Έντασης Προτιμήσεων**: Η μετρική Jaccard αγνοεί την πραγματική τιμή των αξιολογήσεων και ενδιαφέρεται μόνο για την παρουσία ή απουσία αξιολογήσεων. Αυτό σημαίνει ότι χάνεται η πληροφορία για το πόσο πολύ ένας χρήστης προτιμά ή δεν προτιμά μια ταινία σε σχέση με έναν άλλο χρήστη. Σε αντίθεση, η Ευκλείδεια απόσταση και η Συνημίτονο απόσταση λαμβάνουν υπόψη το μέγεθος των προτιμήσεων.
2. **Μη Ευαισθησία στην Κλίμακα**: Δεδομένου ότι η Jaccard εστιάζει στη δυαδική σύγκριση, δεν είναι ευαίσθητη στις διαφορές κλίμακας των αξιολογήσεων. Για παράδειγμα, αν δύο χρήστες έχουν παρόμοιες αξιολογήσεις αλλά σε διαφορετική κλίμακα (π.χ., ο ένας είναι γενικά πιο αυστηρός και δίνει χαμηλότερες βαθμολογίες), η Jaccard δεν θα το λάβει υπόψη.
3. **Ασυμμετρία Πληροφοριών**: Σε περιπτώσεις όπου οι χρήστες έχουν αξιολογήσει πολύ διαφορετικό αριθμό ταινιών, η μετρική Jaccard μπορεί να είναι λιγότερο αξιόπιστη. Για παράδειγμα, ένας χρήστης με πολλές αξιολογήσεις μπορεί να έχει πολύ λίγα κοινά στοιχεία με έναν χρήστη που έχει λίγες αξιολογήσεις, και αυτό μπορεί να οδηγήσει σε υψηλή απόσταση, ακόμα κι αν οι χρήστες είναι ουσιαστικά παρόμοιοι.
4. **Υπολογιστική Αποδοτικότητα**: Ο υπολογισμός της μετρικής Jaccard μπορεί να είναι πιο απαιτητικός σε σχέση με την Ευκλείδεια ή τη Συνημίτονο απόσταση, ειδικά σε πολύ μεγάλα σύνολα δεδομένων, επειδή απαιτεί την υπολογισμό τομών και ενώσεων για κάθε ζεύγος χρηστών.

Συμπέρασμα

Η μετρική Jaccard είναι εξαιρετικά χρήσιμη για τη σύγκριση δυαδικών ή αραιών δεδομένων, αλλά χάνει μέρος της πληροφορίας που παρέχουν οι αριθμητικές τιμές των αξιολογήσεων. Σε πολλές περιπτώσεις, η Ευκλείδεια και η Συνημίτονο απόσταση μπορεί να παρέχουν πιο πλούσια πληροφόρηση για τη σχέση μεταξύ των χρηστών, λαμβάνοντας υπόψη και τις πραγματικές τιμές των αξιολογήσεων.

b) Η μετρική που περιγράφεται μέσω της σχέσης (5) μπορεί να χρησιμοποιηθεί προκειμένου να προσδιοριστεί το σύνολο $Nk(ua)=\{ua(1),ua(2),\cdots,ua(k)\}$ των k πλησιέστερων γειτόνων ενός χρήστη $ua \in UGa, \forall a \in [L]$. Επομένως, για τον εκάστοτε χρήστη ua της εκάστοτε συστάδας UGa μπορούμε να το διάνυσμα τον προσωπικών του προτιμήσεων Rua καθώς και τα διανύσματα των k πλησιέστερων γειτόνων του $Rua(1)$, $Rua(2),\ldots,$ $Rua(\kappa)$ εντός της συστάδας UGa. Στόχος του συγκεκριμένου ερωτήματος είναι να αναπτύξετε ένα πολυστρωματικό νευρωνικό δίκτυο για κάθε συστάδα χρηστών UGa με $\in []$ το οποίο θα προσεγγίζει τις αξιολογήσεις του κάθε χρήστη εντός αυτής μέσω των αξιολογήσεων των k πλησιέστερων γειτόνων του μέσω μιας συνάρτησης της μορφής: $ua=fa(Rua(1),Rua(2),\cdots Rua(k)),\forall a \in UGa,\forall a \in [L]$(6)

Προσδιορισμός των Πλησιέστερων Γειτόνων

Ο κώδικας στο *ClusterModelTrainer.py* χρησιμοποιεί την απόσταση Jaccard, για να δημιουργήσει έναν πίνακα αποστάσεων μεταξύ των χρηστών. Στη συνέχεια, βασιζόμενος σε αυτόν τον πίνακα, ο κώδικας υπολογίζει τους k πλησιέστερους γείτονες για κάθε χρήστη. Αυτό το τμήμα κώδικα βρίσκεται στη μέθοδο *calculate_k_neighbors*.

```
def calculate_k_neighbors(self, k):
    # Ενημερώνουμε τον αριθμό των γειτόνων που θέλουμε να εξετάσουμε
    self._k_nearest_neighbors = np.argsort(self._distance_matrix, axis=1)[:, 1:k+1]
```

Πολυστρωματικό Νευρωνικό Δίκτυο

Η μέθοδος *train_model* του *ClusterModelTrainer* είναι υπεύθυνη για την εκπαίδευση του νευρωνικού δικτύου. Η είσοδος του δικτύου είναι τα διανύσματα των προτιμήσεων του χρήστη καθώς και των k πλησιέστερων γειτόνων του. Το δίκτυο έχει ως στόχο να προβλέψει την αξιολόγηση του χρήστη βασισμένο στις αξιολογήσεις των πλησιέστερων γειτόνων του.

```
def train_model(self, epochs=10): #το epochs είναι συντελεστής, όσο πιο μεγάλος είναι τόσο πιο overfitting μπορεί να γίνουν
    model = self.build_model(len(self.ratings[0]))  # Χρησιμοποιούμε τον αριθμό ταινιών ως input_dim
    print(self.train_data.shape)
    history = model.fit(self.train_data, self.train_labels, epochs=epochs,
                        validation_data=(self.test_data, self.test_labels), verbose=1)
    return history
```

Κατασκευή του Νευρωνικού Δικτύου

Η μέθοδος *build_model* δημιουργεί το πολυστρωματικό νευρωνικό δίκτυο που θα χρησιμοποιηθεί για την εκπαίδευση.

```
def build_model(self, input_dim): #input_dim is number of features,δηλαδή στήλες στον πίνακα με τις αξιολογήσεις
    # Δημιουργία του νευρωνικού δικτύου
    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(input_dim,)),  # Specify the input shape explicitly
        tf.keras.layers.Dense(128, activation='relu', input_dim=input_dim),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(1)  # Προβλέπουμε μία συνεχή τιμή αξιολόγησης
    ])
    model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_absolute_error'])
    return model
```

Αυτό το νευρωνικό δίκτυο:

- Λαμβάνει ως είσοδο τα διανύσματα προτιμήσεων του χρήστη και των πλησιέστερων γειτόνων του.
- Προβλέπει την αξιολόγηση του χρήστη για μια ταινία βασισμένο στις αξιολογήσεις των γειτόνων του.

c) Το σύνολο των χρηστών της κάθε συστάδας μπορεί να διαμεριστεί περεταίρω σε ένα υποσύνολο χρηστών για εκπαίδευση του νευρωνικού δικτύου UGa_{train} *και σε ένα υποσύνολο χρηστών για τον έλεγχο της επίδοσης του νευρωνικού δικτύου* UGa_t *έτσι ώστε* $UGa = UGa_{train} \cup UGa_t$ *με* $UGa_{train} \cap UGa_t = \emptyset$*. Η καταλληλότερη διαμόρφωση των δεδομένων για την εκπαίδευση των νευρωνικών δικτύων αυτού του ερωτήματος θα μπορούσε να αναπαρασταθεί ως εξής: Έστω ότι το σύνολο των χρηστών που μετέχουν στην συστάδα* UGa *δίνεται από την σχέση:* $UGa = \{a,1, a,2, \cdots, a,na\}$ *με* $a = UGa$*. Τότε το σύνολο των διανυσμάτων χαρακτηριστικών και το αντίστοιχο σύνολο των ετικετών θα μπορούσε να οργανωθεί σε έναν πίνακα της μορφής:* $[Rua,1(1) Rua,1(2) \cdots Rua,1(k) Rua,2(1) Rua,2(2) \cdots Rua,2(k)$ ⋮⋮⋮⋮ $Rua,na(1) Rua,na(2) \cdots Rua,na(k)]$ και $[ua,1 ua,2 \vdots ua,na]$

Διαχωρισμός των Δεδομένων σε Εκπαίδευση και Έλεγχο

Στον κώδικα, η μέθοδος *split_data()* της κλάσης *ClusterModelTrainer* είναι αυτή που καλύπτει αυτή τη λειτουργία. Αυτή η μέθοδος χρησιμοποιεί τη βιβλιοθήκη *train_test_split* του *sklearn* για να διαχωρίσει τα δεδομένα των χρηστών σε εκπαιδευτικό και δοκιμαστικό σύνολο.

```python
def split_data(self, test_size=0.2):
    # Διαχωρισμός των δεδομένων σε εκπαιδευτικό και δοκιμαστικό σετ
    self.train_data, self.test_data, self.train_labels, self.test_labels = train_test_split(
        self.ratings, self.ratings, test_size=test_size, random_state=42
    )
```

- *train_data*: Τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση του νευρωνικού δικτύου.
- *test_data*: Τα δεδομένα που θα χρησιμοποιηθούν για τον έλεγχο της επίδοσης του νευρωνικού δικτύου.
- Η αναλογία του εκπαιδευτικού προς το δοκιμαστικό σύνολο ορίζεται με την παράμετρο *test_size*, που έχουμε ορίσει ως 20% των συνολικών δεδομένων.

Διαμόρφωση Δεδομένων για Εκπαίδευση

Για κάθε χρήστη, τα δεδομένα διαμορφώνονται όπως περιγράφεται στον πίνακα της ερώτησης (c). Αυτό επιτυγχάνεται με τη χρήση των διανυσμάτων αξιολογήσεων των πλησιέστερων γειτόνων (όπως υπολογίζονται από τη μέθοδο *calculate_k_neighbors*) και στη συνέχεια το νευρωνικό δίκτυο εκπαιδεύεται χρησιμοποιώντας αυτά τα διανύσματα ως εισόδους.

Στον κώδικα, αυτή η διαδικασία καλύπτεται με το νευρωνικό δίκτυο που αναπτύσσεται στη μέθοδο *train_model()* που ήδη αναλύσαμε στην προηγούμενη απάντηση.

d) Η προσεγγιστική ακρίβεια των παραπάνω νευρωνικών δικτύων μπορεί να μετρηθεί μέσω της μετρικής του μέσου απόλυτου σφάλματος ανάμεσα στις πραγματικές και τις εκτιμώμενες αξιολογήσεις των χρηστών. Να παρουσιάσετε πίνακες των αποτελεσμάτων σας τόσο για την ακρίβεια εκπαίδευσης όσο και για την ακρίβεια ελέγχου για κάθε συστάδα χρηστών.

Τρέχοντας τον κώδικα τα αποτελέσματα που λάβαμε για την ακρίβεια της εκπαίδευσης και του ελέγχου για κάθε συστάδα όπως προέκυψαν είναι τα παρακάτω:

Μέσο Απόλυτο Σφάλμα (Εκπαίδευση):

[

2.5958468914031982,

1.984748363494873,
1.9682090282440186,
1.9822108745574951,
1.8626470565795898,
1.906843900680542,
1.9328279495239258,
1.8936686515808105,
1.902924656867981,
1.904740571975708
]

Μέσο Απόλυτο Σφάλμα (Έλεγχος):

[
2.313577175140381,
2.292059898376465,
1.9441765546798706,
1.9677627086639404,
2.076704502105713,
2.0017974376678467,
1.9834747314453125,
2.0636134147644043,
2.002134084701538,
2.032474994659424
]

Ανάλυση των Αποτελεσμάτων

1. **Μέσο Απόλυτο Σφάλμα (Εκπαίδευση)**:

Οι τιμές κυμαίνονται μεταξύ 1.86 και 2.59, δείχνοντας ότι το μοντέλο σταδιακά βελτιώνεται κατά την εκπαίδευση, αλλά το σφάλμα παραμένει πάνω από 1.86, κάτι που υποδεικνύει ότι το μοντέλο προσπαθεί να προσαρμοστεί στα δεδομένα εκπαίδευσης.

2. **Μέσο Απόλυτο Σφάλμα (Έλεγχος)**:

Οι τιμές κυμαίνονται μεταξύ 1.94 και 2.31. Παρατηρούμε ότι το σφάλμα στο δοκιμαστικό σύνολο είναι γενικά υψηλότερο από το σφάλμα στο εκπαιδευτικό σύνολο, με εξαίρεση το 3ο και 4ο epoch, όπου το σφάλμα στο validation είναι μικρότερο από αυτό της

εκπαίδευσης, κάτι που μπορεί να υποδηλώνει τυχαίες διακυμάνσεις στα δεδομένα.

3. **Ρυθμός Μεταβολής**:

Ο ρυθμός μεταβολής του σφάλματος κατά την εκπαίδευση (mean_dt) είναι -0.0768, ενώ κατά τον έλεγχο (val_dt) είναι -0.0312. Αυτό υποδεικνύει ότι το μοντέλο βελτιώνεται κατά την εκπαίδευση, αλλά το σφάλμα στον έλεγχο δεν μειώνεται τόσο πολύ όσο στην εκπαίδευση, κάτι που μπορεί να δείχνει ότι το μοντέλο έχει αρχίσει να υπερ προσαρμόζεται στα δεδομένα εκπαίδευσης (overfitting).

Github:https://github.com/tassosant/data_analysis_master.git