Ανάκτηση Πληροφορίας

Project 2022-2023

Στυλιανάκης Στυλιανός	1059713	steliostylian@gmail.com
Κονταράκης Ιωάννης Γεώργιος	1067375	kogi_20010@hotmail.com

Link to GitHub

Πίνακας περιεχομένων

Περιβάλλον υλοποίησης

Ως γλώσσα υλοποίησης επιλέχθηκε η **python**, καθώς υπάρχουν πολλές χρήσιμες βιβλιοθήκες για την εργασία.

Βιβλιοθήκες που χρησιμοποιήθηκαν

`pandas`

Βασική βιβλιοθήκη διαχείρισης δεδομένων. Τα περισσότερα δεδομένα μας φορτώνονται, επεξεργάζονται και ανακτούνται από **DataFrames**, τη βασική δομή δεδομένων των pandas.

`elasticsearch`

Χρησιμοποιήθηκε για τη σύνδεση με την Elasticsearch, καθώς και την πραγματοποίηση των queries, μέσω REST API calls.

`sklearn`

Βιβλιοθήκη με πολλές χρήσιμα εργαλεία για υλοποίηση clustering, καθώς και νευρωνικών δικτύων. Χρησιμοποιήθηκε κυρίως στο ερώτημα 3.

`kmodes`

Βιβλιοθήκη για clustering, η οποία χρησιμοποιεί και την **sklearn** (σε συνδυασμό με άλλες βιβλιοθήκες). Συγκεκριμένα, από αυτήν τη βιβλιοθήκη χρησιμοποιήσαμε την κλάση **KPrototypes** για να υλοποιήσουμε το clustering.

`matplotlib, pyplot, seaborn`

Βιβλιοθήκες για σχεδιασμό και εμφάνιση γραφημάτων.

`re`

Βιβλιοθήκη αναγνώρισης κανονικών εκφράσεων, χρησιμοποιήθηκε στον καθαρισμό των datasets.

Ερώτημα 1 – ElasticSearch

Σύνδεση με την Elastic, δημιουργία δείκτη & εισαγωγή βιβλίων

Ξεκινώντας, αρχικοποιούμε ένα instance της κλάσης `Elasticsearch` με τα απαραίτητα στοιχεία για τη σύνδεση στο cluster μας. Την πρώτη φορά που θα τρέξουμε την Elasticsearch, πρέπει να δημιουργήσουμε ένα index στο οποίο θα προσθέσουμε τα βιβλία, όπως φαίνεται παρακάτω.

```python
def main():
    ############################## ELASTICSEARCH ##############################

    # Connect to the ElasticSearch cluster
    es = Elasticsearch(
        "http://localhost:9200",
        ssl_assert_fingerprint=CERT_FINGERPRINT,
        basic_auth=("elastic", ES_PASSWORD)
        )

    # Create a new index in ElasticSearch called books
    print("Creating books index...")
    elastic.createIndex(es, idx_name="books")

    # Insert a sample of data from dataset into ElasticSearch
    book_count = elastic.insertData(es)[0]["count"]
    print(f"Inserted {book_count} books into index.")
```
main.py

Με τη μέθοδο `createIndex` δημιουργούμε το books index, ορίζοντας τις ιδιότητες των δεδομένων που θα εισάγουμε. Σε σχόλιο υπάρχει η μέθοδος με την οποία μπορούμε να διαγράψουμε το index που έχουμε δημιουργήσει για να το ξαναφτιάξουμε από την αρχή.

```python
def createIndex(es: Elasticsearch, idx_name: str = "books") -> None:
  """Creates a new book index, deleting
  any pre-existing with the same name."""

  # Define the mappings of the books index.
  mappings = {
    "properties": {
      "isbn": {"type": "text", "analyzer": "keyword"},
      "book_title": {"type": "text", "analyzer": "english"},
      "book_author": {"type": "text", "analyzer": "standard"},
      "year_of_publication": {"type": "integer"},
      "publisher": {"type": "text", "analyzer": "standard"},
      "summary": {"type": "text", "analyzer": "english"},
      "category": {"type": "text", "analyzer": "standard"}
    }
  }

  # Delete pre-existing index with the same name
  #es.indices.delete(index=idx_name)
  # Create a new index named idx_name with the defined mappings
```

```
  es.indices.create(index=idx_name, mappings=mappings)
```
elastic.py

Με τη μέθοδο `insertData` εισάγουμε τα βιβλία στη βάση. Επιλέξαμε να τα εισάγουμε όλα μαζί (in bulk) με μία κλήση στην Elastic, η οποία περιέχει λίστα με όλα τα βιβλία. Η μέθοδος επιστρέφει την απάντηση που πήραμε από την Elastic μετά την εισαγωγή των βιβλίων, όπου φαίνεται πόσα βιβλία υπάρχουν.

```
def insertData(es: Elasticsearch) -> str:
  """Parses the data of a specified csv file and
  inserts the data into Elasticsearch. Returns
  the number of entries after insertion."""

  # Parse the CSV dataset
  dataframe = pd.read_csv(BOOKS).dropna().reset_index()

  # Create a list containing the parsed rows from the CSV file
  bulk_data = []
  for i,row in dataframe.iterrows():
      bulk_data.append(
          {
              "_index": "books",
              "_id": i,
              "_source": {
                "isbn": row["isbn"],
                "book_title": row["book_title"],
                "book_author": row["book_author"],
                "year_of_publication": row["year_of_publication"],
                "publisher": row["publisher"],
                "summary": row["summary"],
                "category": row["category"]
              }
          }
      )
  # Bulk insert the rows into ElasticSearch
  bulk(es, bulk_data)

  # Refresh the books index and return the number of items in it.
  es.indices.refresh(index="books")
  resp = es.cat.count(index="books", format="json")
  return resp
```
elastic.py

Πραγματοποίηση ερωτημάτων

Για την πραγματοποίηση ερωτημάτων, ο χρήστης πρέπει να εισάγει έναν αλφαριθμητικό όρο, τον οποίο θα αναζητήσει η Elastic στα βιβλία που της έχουμε φορτώσει στη βάση, καθώς και το αναγνωριστικό του.

```
    # Input query parameters and make query to ES
    search_string = input("Enter search string:")
    user_id = _input("Enter your user ID (must be an integer): ", int)
    es_reply = elastic.makeQuery(es, search_string)
```
main.py

Τα ερωτήματα γίνονται μέσω της μεθόδου makeQuery. Για κάθε ερώτημα, ζητάμε στην Elastic να αναζητήσει τους όρους σε πολλαπλά πεδία των εγγραφών, πιο συγκεκριμένα στους τίτλους των βιβλίων, καθώς και στην περιγραφή τους. Όπως φαίνεται στο σώμα του ερωτήματος στη γραμμή των πεδίων, αποφασίσαμε να δώσουμε πιο μεγάλο βάρος στον τίτλο του βιβλίου (1.5 αντί για 1 συγκεκριμένα), παρά στην περίληψη. Αυτήν την απόφαση την πήραμε βασισμένοι στην υπόθεση ότι, αν οι όροι αναζήτησης υπάρχουν στον τίτλο, ουσιαστικά είναι εγγυημένο ότι περιλαμβάνονται στο **βασικό** θέμα του βιβλίου, σε αντίθεση με την περίληψη.

```
def makeQuery(es: Elasticsearch, search_string: str) -> tuple:
  """Creates a query and returns Elasticsearch's answer."""

  # Create query body using inputs. Using multi_match we check multiple fields
  # of an entry for the given search string and the final score is calculated
  # by adding the (weighted) score of the fields.
  query_body = {
    "query": {
        "multi_match": {
            "query": search_string,
            "type": "most_fields",
            "fields": ["book_title^1.5", "summary"]
        }
    }
  }

  # Make the query to ElasticSearch
  es_reply = es.search(
    index = "books",
    body = query_body,
    size = 10_000
  )
```

```
  return es_reply["hits"]
```
elastic.py

Υπολογισμός συνδυαστικών βαθμολογιών

Μόλις λάβουμε την απάντηση της Elastic, υπολογίζουμε τη συνδυαστική βαθμολογία των βιβλίων συνυπολογίζοντας τις πιθανές βαθμολογίες που έχουν κάνει οι χρήστες στα σχετικά βιβλία.

```
    # Get the combined scores for all replies
    print("\nCalculating combined scores...")
    combined_scores = functions.calculateCombinedScores(es_reply, user_id)
    combined_scores.to_csv(SCORES_NO_CLUST, index=False)
```
main.py

Η μέθοδος `calculateCombinedScores` χρησιμοποιείται τόσο για το 1ο, όσο και για το 2ο ερώτημα, μετά το clustering. Για αυτόν το λόγο, έχει κάποια προαιρετικά ορίσματα τα οποία μας χρησιμεύουν μόνο στο 2ο ερώτημα. Τα βασικά ορίσματα που χρειάζονται και για τα 2 ερωτήματα, είναι η απάντηση της Elastic και το αναγνωριστικό του χρήστη. Τελικά, η μέθοδος επιστρέφει ένα DataFrame με την ανανεωμένη συνδυαστική βαθμολογία.

```
def calculateCombinedScores(es_reply: dict, user_id: int, use_cluster_ratings:\
    bool = False, avg_clust_ratings: pd.DataFrame = None,\
                    cluster_assigned_users: pd.DataFrame = None) -> pd.DataFrame:
```
functions.py

Αρχικά, η μέθοδος αποθηκεύει το `max_score`, όλα τα βιβλία που επέστρεψε η Elastic σε μία λίστα από λεξικά και φορτώνει όλες τις αξιολογήσεις του χρήστη σε ένα DataFrame, χρησιμοποιώντας τη μέθοδο `getUserRatings`.

```
    books = es_reply["hits"]
    max_score = es_reply["max_score"]
    user_ratings = getUserRatings(user_id)
                  print(f"User {user_id} has rated {len(user_ratings)} book(s).")
```
functions.py

Η μέθοδος `getUserRatings` απλώς διαβάζει το CSV με όλα τα Book Ratings και επιστρέφει μόνο τις γραμμές που περιέχουν το `user_id` που της περνάμε.

```
def getUserRatings(user_id: int, filename: str = RATINGS) -> pd.DataFrame:
    """Read ratings CSV and return specified user's ratings."""

    users_ratings_df = pd.read_csv(filename)
    return  users_ratings_df.loc[users_ratings_df["uid"] == user_id]
```
functions.py

Στη συνέχεια, αρχικοποιεί μία άδεια λίστα όπου θα μπουν τα βιβλία αφού υπολογιστεί η συνδυαστική βαθμολογία τους. Έπειτα, εάν η μέθοδος έχει

κληθεί με το use_cluster_ratings=True (όπως καλείται στο 2ο ερώτημα δηλαδή), ανακτάται από το αντίστοιχο dataframe η συστάδα στην οποία ανήκει ο χρήστης, εάν αυτός υπάρχει στο αρχείο BX-Users.csv. Σε περίπτωση που δεν υπάρχει, εκτυπώνεται αντίστοιχο μήνυμα.

```
    # List to be filled with book entries
    # and their combined scores
    books_list = []
    users_cluster = -1

    # Get user's cluster
    if use_cluster_ratings:
        try:
            users_cluster = cluster_assigned_users["Cluster"].\
                loc[cluster_assigned_users["User_ID"] == user_id].iloc[0]
            print(f"User {user_id} belongs in cluster {users_cluster}")
        except:
            print(f"User {user_id} doesn't exist in database.")
            print("Scores with clustering will be the same as previous scores.")
```
functions.py

Το κύριο κομμάτι της μεθόδου αυτής είναι η βασική της επανάληψη, στην οποία υπολογίζεται η συνδυαστική βαθμολογία κάθε βιβλίου και δημιουργούνται εγγραφές βιβλίων. Οι εγγραφές αυτές περιέχουν όλες τις πληροφορίες κάθε βιβλίου, καθώς και την υπολογισμένη βαθμολογία τους. Οι εγγραφές αυτές προστίθενται στην books_list.

Αρχικά, προσπαθούμε να ανακτήσουμε τη βαθμολογία του χρήστη για το συγκεκριμένο βιβλίο. Σε περίπτωση που δεν το έχει βαθμολογήσει και δε χρησιμοποιούμε clustering, πριν υπολογιστεί η συνδυαστική βαθμολογία, θέτουμε τη βαθμολογία του χρήστη ίση με DEFAULT_RATING, σταθερά η οποία έχει οριστεί στην αρχή του αρχείου.

Ως τιμή για αυτή τη σταθερά επιλέξαμε να είναι το 5, η μέση τιμή των πιθανών βαθμολογιών δηλαδή. Επιλέξαμε την τιμή αυτή επειδή θέλουμε τα βιβλία που έχει βαθμολογήσει αρνητικά ο χρήστης (δηλαδή με βαθμολογία κάτω από 5) να εμφανίζονται πιο κάτω στην κατάταξη από ότι αυτά που δεν έχει βαθμολογήσει ακόμα. Αντίστοιχα, τα βιβλία στα οποία έχει βάλει θετική βαθμολογία, προωθούνται πιο ψηλά στην κατάταξη.

```
    # Iterate through the documents of the ES reply
    for book in books:
        isbn = book["_source"]["isbn"]
        norm_es_score = 10*book["_score"]/max_score

        # User has rated book
```

```
        try:
            # Get user's book rating and typecast it to float (from string)
            user_rating = float(user_ratings_df.\
                loc[user_ratings ["isbn"] == isbn]["rating"].iloc[0])
        # User has not rated this book
        except:
            # User doesn't exist in the database or function has
            # been called with arg use_cluster_ratings = False
            if users_cluster == -1:
                user_rating = DEFAULT_RATING

            else:
                user_rating = getAvgClusterRating(
                    users_cluster, isbn, avg_clust_ratings)

        # Combined score will be calculated using combinedScoreFunc
        score = combinedScoreFunc(norm_es_score, user_rating)
        # Create a new book entry as a list
        new_book = [
            score, user_rating, norm_es_score, isbn,
            book['_source']["book_title"],
            book['_source']["book_author"],
            book['_source']["year_of_publication"],
            book['_source']["publisher"],
            book['_source']["summary"],
            book['_source']["category"]
            ]
        books_list.append(new_book)
```

functions.py

Για τον υπολογισμό της συνδυαστικής βαθμολογίας, χρησιμοποιείται η μέθοδος `combinedScoreFunc`, η οποία παίρνει σαν είσοδο το score της Elasticsearch (κλιμακωμένο στο διάστημα [0, 10]), καθώς και τη βαθμολογία του χρήστη.

Στη βαθμολογία του χρήστη έχουμε προσθέσει ένα βάρος, ώστε να έχει μεγαλύτερη σημασία από ότι το σκορ της Elastic. Ο λόγος που δε σκεφτήκαμε την αντίθετη περίπτωση είναι επειδή η Elastic επιστρέφει μόνο σχετικά βιβλία στα οποία έχει βρει τους όρους αναζήτησης. Πειραματιστήκαμε λίγο με τη σταθερά αυτή και καταλήξαμε στο βάρος 1.25 για τη βαθμολογία του χρήστη.

Η συνδυαστική βαθμολογία προκύπτει από το σταθμισμένο άθροισμα των 2 εισόδων, κλιμακωμένο στο διάστημα [0, 10] και επιστρέφεται από τη μέθοδο.

```
def combinedScoreFunc(norm_es_score: float, user_rating: float) -> float:
```

```
    """Function that accepts as inputs the elastic search score of a book,
    as well as the user's rating of the book and returns a combined score."""

    # Add the normalized and weighted scores and scale them in the range [0, 10]
    return (USER_R_WEIGHT * user_rating + norm_es_score) / (1 + USER_R_WEIGHT)
```
functions.py

Αφού τελειώσει η επανάληψη και έχουμε προσθέσει όλες τις εγγραφές των σχετικών βιβλίων στην books_list, δημιουργούμε ένα DataFrame με τις εγγραφές αυτές, προσθέτοντας και τα ονόματα των αντίστοιχων στηλών τους. Τέλος, τα ταξινομούμε και επιστρέφουμε μόνο τις 10% καλύτερες αντιστοιχίσεις, σύμφωνα με τις συνδυαστικές τους βαθμολογίες.

```
    # Create a new dataframe from books_list and sort it by score
    best_matches = pd.DataFrame(data=books_list, columns=[
        "score", "user_rating", "es_scaled_score", "isbn", "book_title",
        "book_author", "year_of_publication", "publisher", "summary",
        "category"]).sort_values(by="score", ascending=False)

    # Only keep the best 10% documents
    return best_matches.head(len(best_matches.index)//10)
```
functions.py

Ερώτημα 2 – Clustering

Προεπεξεργασία του users dataset

Πριν ξεκινήσουμε το clustering, πρέπει να γίνει μία προεπεξεργασία του Users dataset. Για αυτόν το σκοπό χρησιμοποιούμε τη μέθοδο `processUsersCSV`.

```
    # Create a CSV containing users sorted by their country
    # Entries that have a high chance of being fake are ignored
    print("Creating processed users CSV...")
    processed_users = functions.processUsersCSV()
    processed_users.to_csv(PROC_USERS, index=False)
```
main.py

Αρχικά, η μέθοδος διαβάζει το αρχείο BX-Users.csv, το φορτώνει σε ένα dataframe και διατρέχει τις γραμμές του, εξάγοντας το uid του χρήστη, την ηλικία του, τη χώρα του και το μοναδικό id της. Για τον κάθε χρήστη, αυτές οι τιμές αποθηκεύονται σε μία λίστα η οποία με τη σειρά της προστίθεται στη λίστα χρηστών users. Τελικά, δημιουργείται ένα dataframe με τους χρήστες, ορίζοντας τα κατάλληλα ονόματα για την κάθε στήλη.

Σε περίπτωση που ο χρήστης δεν έχει συμπληρώσει ηλικία ή ηλικία που έχει συμπληρώσει δεν ανήκει σε ένα φυσιολογικό διάστημα [0, 120], του

αναθέτουμε μία καινούρια τιμή. Επιλέξαμε αυτή η τιμή να είναι η μέση τιμή του διαστήματος, δηλαδή 60. Άλλη ανάθεση που θα μπορούσε να γινόταν θα ήταν να διαλέγαμε τυχαία έναν αριθμό στο διάστημα αυτό. Σε κάθε περίπτωση, δεν μπορούμε να αφήσουμε κενή την ηλικία, καθώς ο αλγόριθμος του clustering δε δέχεται μη αριθμητικές τιμές.

Η χώρα του εξάγεται με τη χρήση μίας κανονικής έκφρασης η οποία ουσιαστικά διαβάζει όλους τους χαρακτήρες μετά το τελευταίο κόμμα του αλφαριθμητικού location.

Το μοναδικό id της κάθε χώρας είναι ουσιαστικά η σειρά εμφάνισής της στο dataset μας, με εξαίρεση της κενές εγγραφές. Για κάθε εγγραφή που έχει κενή χώρα, το id της χώρας υπολογίζεται ως συνάρτηση του uid, αυξημένο κατά 1000. Το 1000 επιλέχθηκε επειδή, τουλάχιστον για το dataset μας, εγγυάται ότι δε θα υπάρξουν συγκρούσεις με τα ids άλλων χωρών.

Η στήλη uid αποθηκεύεται όπως είναι, αφού γίνει μετατροπή του αλφαρηθμιτικού σε ακέραιο.

```
def processUsersCSV() -> pd.DataFrame:
    """Extract vital information from BX-Users csv and save it to a new CSV."""
    # Create empty users dictionary
    users = []
    countries = []

    # Iterate BX-Users.csv rows
    for entry in pd.read_csv(USERS).iterrows():
        # Cast age to int
        uid = int(entry[1][0])
        # Get location
        location = entry[1][1]
        # Try to cast age to int. If it's empty or
        # out of bounds, set it to the default age
        try:
            age = int(entry[1][2])
            if age > MAX_AGE or age < 0:
                age = DEF_AGE
        except:
            age = DEF_AGE

        # Extract country from location string
        country = re.findall(r"[\s\w+]+$", location)
        # If country is empty, set country_id to 1000+uid, so that users with
        # incomplete countries do not belong in the same "country". Essential
        # for clustering! Also, 500 is arbitrarily chosen butis empirically
```

```
        # higher than the count of normal countries, so there are no conflicts.
        if not country:
            country = ""
            country_id = 1000 + uid

        else:
            country = country[0][1:]

        # If country isn't already in the countries list,
        # its index will be equal to the length of countries
        if country not in countries:
            country_id = len(countries)
            countries.append(country)
        # Otherwise, its index is equal to the index of the country in countries
        else:
            country_id = countries.index(country)
        users.append([uid, age, country, country_id])

    users_df = pd.DataFrame(
        users, columns=["User_ID", "Age", "Country", "Country_ID"]
    )

    return users_df
```

functions.py

Αλγόριθμος clustering

Το πρόβλημα

Στην εργασία, ζητείται ως αλγόριθμος ομαδοποίησης να χρησιμοποιηθεί ο Kmeans. Παρόλα αυτά, ο απλός Kmeans δέχεται **μόνο** αριθμητικές ιδιότητες (features). Πιο συγκεκριμένα, ο αλγόριθμος αρχικοποιεί τυχαία k κεντροειδή, τα οποία αποτελούν αρχικά το κέντρο της κάθε συστάδας. Στη συνέχεια, υπολογίζει τις ευκλείδειες αποστάσεις των σημείων (δεδομένων μας) από τα κεντροειδή, αναθέτοντας το κάθε σημείο στο κοντινότερο κεντροειδές. Στο τέλος του βήματος, ξαναυπολογίζονται τα κεντροειδή ως το μέσο των σημείων τα οποία ανήκουν σε αυτό. Ο αλγόριθμος τερματίζει όταν τα κεντροειδή δεν μετακινούνται.

Λύση #1 – kMeans

Θα μπορούσαμε να χρησιμοποιήσουμε τον kMeans από τη βιβλιοθήκη scikit-learn, κι ας μην θεωρούμε αρμόζει στα δεδομένα μας. Παρόλα αυτά, θεωρήσαμε ότι υπάρχουν πιο σωστές λύσεις για τα ζητούμενα της εργασίας.

Λύση #2 – Custom kMeans with geolocation from API

Θα μπορούσαμε να χρησιμοποιήσουμε κάποιο API (πχ το Geocoding της Google) ώστε να μετατρέπουμε τις Πόλεις-Χώρες σε συντεταγμένες και στη συνέχεια να προγραμματίζαμε έναν αλγόριθμο για kMeans χωρίς τη χρήση κάποιας έτοιμης βιβλιοθήκης (πχ scikit-learn). Ο αλγόριθμος αυτός θα εκτελούσε τον kMeans με συνυπολογισμό της χωρικής και ηλικιακής διαφοράς, επαναπροσδιορίζοντας το κέντρο της κάθε συστάδας. Ενώ ίσως είναι η πιο σωστή λύση, είναι εκτός των ορίων της εργασίας.

Λύση #3 - kPrototypes

Απλοποιώντας λίγο το πρόβλημα, θεωρούμε πως η ιδιότητα της χώρας δεν αποτελεί αριθμητική ιδιότητα, αλλά κατηγορηματική. Αυτό σημαίνει πως δεν έχει νόημα να χρησιμοποιήσουμε τις αποστάσεις των ids των χωρών ως είσοδο για τον αλγόριθμο ομαδοποίησης. Για παράδειγμα, δεν έχει νόημα να πούμε ότι η χώρα με id 50 έχει μεγαλύτερη απόσταση με τη χώρα με id 1 από ότι έχει με τη χώρα 49. Για το λόγο αυτό, τη χρησιμοποιούμε ως κατηγορηματική ιδιότητα, το οποίο σημαίνει ότι, κατά την ομαδοποίηση, μας ενδιαφέρει μόνο η ισότητα και όχι η απόσταση! Δηλαδή, από τη στιγμή που δεν έχουν το ίδιο id, ο αλγοριθμός θεωρεί ότι όλες οι χώρες «ισαπέχουν».

Ο αλγόριθμος **Kprototypes** είναι ουσιαστικά μία βελτίωση των αλγορίθμων kMeans και K-Mode (ο οποίος χρησιμοποιείται για την ομαδοποίηση κατηγορηματικών δεδομένων) που δέχεται τόσο αριθμητικές, όσο και κατηγορηματικές ιδιότητες.

Η είσοδος του αλγορίθμου δημιουργείται με τη μέθοδο `getClusteringInput`, η οποία διαβάζει το dataset των επεξεργασμένων χρηστών, κρατάει τις χρήσιμες για εκείνη στήλες, δηλαδή το Country_ID και το Age. Συγκεκριμένα, το age περνάει από τυποποίηση (standardization), μεταφέρονται δηλαδή οι τιμές του στο διάστημα [-1, 1].

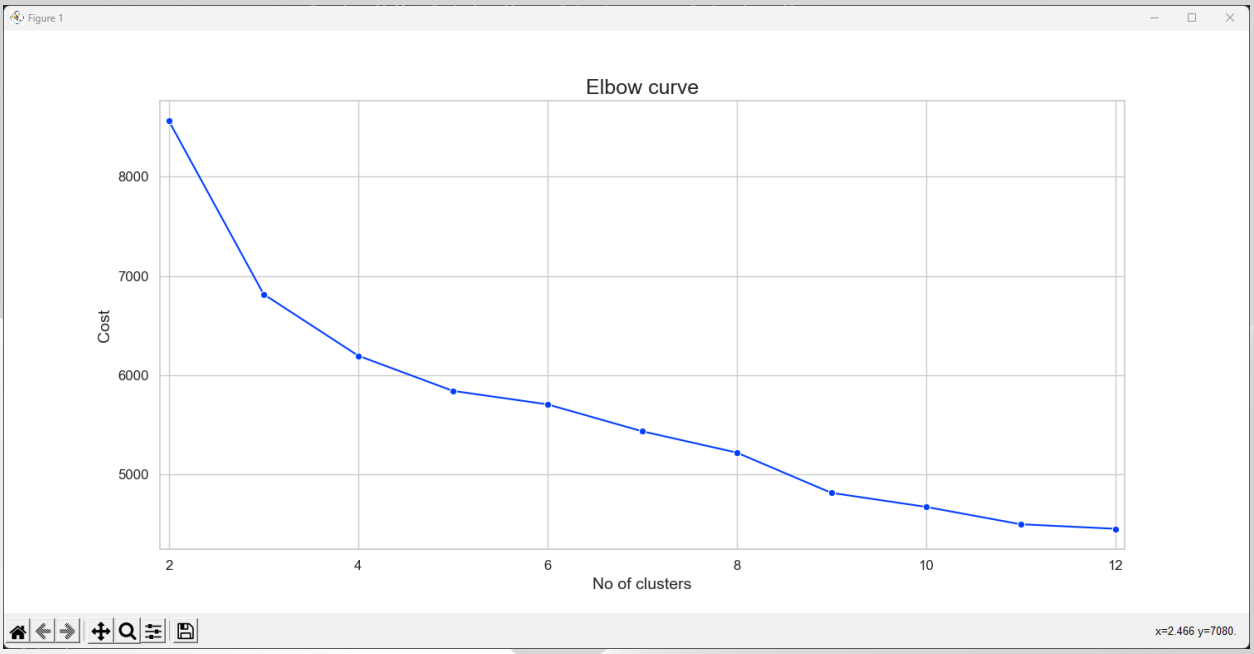
Για να αποφασίσουμε τον αριθμό των συστάδων χρησιμοποιήσαμε τη μέθοδο του αγκώνα, την οποία σχεδιάζομε γραφικά με τη μέθοδο `plot_elbow_curve`.

```
    # Plot elbow curve to help determine optimal number of clusters
    print("\nPlotting elbow curve...")
    clustering.plot_elbow_curve(2, 12, 10_000, processed_users)
    k = _input("Choose number of clusters to use: ", int)
```
main.py

Η μέθοδος τρέχει τον αλγόριθμο σε ένα μικρό υποσύνολο, με μεταβαλλόμενο αριθμό συστάδων k, από 2 μέχρι 12, αποθηκεύοντας το κόστος κάθε τιμή του k. Στη συνέχεια εμφανίζουμε γραφικά τις τιμές αυτές και αποφασίζουμε την τιμή του k για την τελική εκτέλεση του αλγορίθμου.

```python
def plot_elbow_curve(start: int, end: int, sample_size: int,
    proc_users: pd.DataFrame) -> None:
    """Plots elbow curve. Used for optimizing K."""

    data = getClusteringInput(proc_users)
    if sample_size > 0 and sample_size < len(data):
        data = data[:sample_size]

    categorical_index = [1]
    no_of_clusters = list(range(start, end+1))
    cost_values = []
    threads = min(PC, cpu_count())
    print(f"Number of available threads: {cpu_count()}")
    print(f"Starting testing using {threads} threads...")

    for k in no_of_clusters:
        print(f"Testing with {k} clusters...")
        test_model = KPrototypes(
            n_clusters=k, init='Huang', n_init=N_INIT, n_jobs=threads
        )
        test_model.fit_predict(data, categorical=categorical_index)
        cost_values.append(test_model.cost_)

    seaborn.set_theme(style="whitegrid", palette="bright", font_scale=1.1)

    plt.figure(figsize=(15, 7))
    ax = seaborn.lineplot(
        x=no_of_clusters, y=cost_values, marker="o", dashes=False
    )
    ax.set_title('Elbow curve', fontsize=18)
    ax.set_xlabel('No of clusters', fontsize=14)
    ax.set_ylabel('Cost', fontsize=14)
    ax.set(xlim=(start-0.1, end+0.1))
    plt.plot()
    plt.show()
```

clustering.py



Σύμφωνα με το παραπάνω γράφημα, καλές τιμές για το k φαίνεται να είναι οι 3, 5 και ίσως και 9.

Αφού επιλεχθεί από τον χρήστη ο αριθμός των συστάδων, εκτελείται η μέθοδος kPrototypes, μετρώντας το χρόνο εκτέλεσής της.

```
    # Run and time k-Prototypes
    print("Starting clustering...")
    start_time = timeit.default_timer()
    cluster_assigned_users = clustering.kPrototypes(k, processed_users)
    elapsed_time = timeit.default_timer() - start_time
    cluster_assigned_users.to_csv(CLUSTER_ASSIGNED_USERS, index=False)
    print(f"Clustering with {k} clusters took {int(elapsed_time//60)}\
        minutes and {int(round(elapsed_time % 60, 0))} seconds.")
```

main.py

Αντίστοιχος κώδικας με το `plot_elbow_curve` περιέχεται και στη βασική μέθοδο του clustering `kPrototypes`, με τη διαφορά φυσικά ότι αυτή τη φορά τρέχει σε όλο το dataset μας. Ο kPrototypes δεν υποστηρίζει επιτάχυνση κάρτας γραφικών όπως ο κλασικός kMeans της βιβλιοθήκης `scikit-learn`, παρόλα αυτά έχει οριστεί η παράμετρος `n_jobs=threads`, ώστε να παραλληλοποιηθεί όσο γίνεται, έστω κι αν εκτελείται στον επεξεργαστή. Παρόλα αυτά, η εκτέλεσή του διαρκεί σημαντικά περισσότερο από όσο του απλού kMeans και για αυτό αποθηκεύουμε την έξοδό του στο ενδιάμεσο αρχείο Cluster-Assigned-Users.csv.

```
def kPrototypes(k: int, proc_users: pd.DataFrame) -> pd.DataFrame:
    """Runs k-Prototypes algorithm for Users, placing them
```

```
    in one of k clusters."""

    categorical_features_idx = [1]
    mark_array = getClusteringInput(proc_users)
    threads = min(PC, cpu_count())
    print(f"Number of available threads: {cpu_count()}")
    print("Starting k-Prototypes using {threads} threads...")
    test_model = KPrototypes(
        n_clusters=k, verbose=2, init='Huang', n_init=N_INIT, n_jobs=threads
    )
    clusters = test_model.fit_predict(
        mark_array, categorical=categorical_features_idx
    )
    # Add Cluster column to dataframe
    proc_users['Cluster'] = list(clusters)

    return proc_users
```
clustering.py

Υπολογισμός μέσων βαθμολογιών συστάδων

Αφού χωρίσουμε τους χρήστες σε ομάδες, υπολογίζουμε τη μέση βαθμολογία της κάθε ομάδας για κάθε βιβλίο που έχει αξιολογίσει με τη μέθοδο `createAvgClusterRatings`.

```
    # Create a dataframe containing the average
    # rating of every book per cluster
    print("Creating average cluster ratings CSV...")
    avg_clust_ratings = functions.createAvgClusterRatings(cluster_assigned_users)
    avg_clust_ratings.to_csv(AVG_CLUSTER_RATINGS)
```
main.py

Η μέθοδος αυτή αρχικά συγχωνεύει τις αξιολογήσεις χρηστών μαζί με το επεξεργασμένο dataframe των χρηστών που περιέχει τις συστάδες στις οποίες ανήκουν. Στη συνέχεια, κρατάει μόνο τις απαραίτητες στήλες isbn, Cluster και rating, ομαδοποιεί τις εγγραφές ανά βιβλίο και συστάδα και υπολογίζει το μέσο όρο της κάθε ομαδοποίησης. Τέλος, επιστρέφει το δημιουργημένο dataframe που περιέχει τις μέσες βαθμολογίες συστάδων, το οποίο αποθηκεύεται στο αντίστοιχο ενδιάμεσο αρχείο.

```
def createAvgClusterRatings(
        cluster_assignement_df: pd.DataFrame
    ) -> pd.DataFrame:
    """Function that accepts as input the cluster assignement DataFrame and
    restores User IDs to it. Then, it combines this DF with the Book-Ratings
    CSV, averaging out the book ratings per cluster. The combined DF contains
```

```
    the columns isbn, cluster and rating and is finally returned."""

    # Open book ratings CSV in read mode
    books_ratings_df = pd.read_csv(RATINGS)
    # Merge the two DataFrames on UIDs
    result = pd.merge(right=books_ratings_df, left=cluster_assignement_df,\
        how="left", left_on="User_ID", right_on="uid", validate="one_to_many")
    # Drop useless columns
    result.drop(
        ["uid", "User_ID", "Country", "Country_ID", "Age"], axis=1, inplace=True)
    # Group ratings by isbn and Cluster and sort resulting DataFrame
    avg_cluster_ratings = result.groupby(["isbn", "Cluster"]).mean()

    return avg_cluster_ratings
```
functions.py

Υπολογισμός συνδυαστικών βαθμολογιών με συμπλήρωση κενών βαθμολογιών από το μέσο όρο συστάδων

Για τη βελτίωση των επιστρεφόμενων βιβλίων, όπου δεν υπάρχει βαθμολογία χρήστη στο 2ο ερώτημα συμπληρώνεται από τη μέση βαθμολογία της συστάδας του χρήστη, όπου αυτή υπάρχει. Αυτό επιτυγχάνεται με την ίδια μέθοδο που χρησιμοποιείται και στο πρώτο ερώτημα `calculateCombinedScores`, η οποία όμως πλέον καλείται με τα κατάλληλα ορίσματα, όπως φαίνεται παρακάτω:

```
    # Get the re-calculated document scores by using
    # user's cluster's average ratings to "rate" unrated books
    print("Re-calculating combined scores using user's cluster's average book ratings...")
    combined_scores_clusters_df = functions.calculateCombinedScores(es_reply, user_id,\
        use_cluster_ratings=True, avg_clust_ratings = avg_clust_ratings,\
            cluster_assigned_users = cluster_assigned_users)
    combined_scores_clusters_df.to_csv(SCORES_W_CLUST, index=False)
```
main.py

Ερώτημα 3 – Νευρωνικό δίκτυο

Για το 3ο ερώτημα, μας ζητάται να εκπαιδεύσουμε ένα νευρωνικό δίκτυο το οποίο θα προβλέπει τις μέσες βαθμολογίες συστάδων για τα βιβλία που δεν έχουν βαθμολογηθεί από αυτές, χρησιμοποιώντας τις περιλήψεις τους. Πριν

όμως περαστούν στο δίκτυό μας, οι περιλήψεις θα πρέπει να υποστούν μία προ-επεξεργασία.

Word Embeddings

Για τη διανυσματοποίηση των περιλήψεων με τη χρήση Word Embedding, δοκιμάσαμε διάφορες μεθόδους και εν τέλει καταλήξαμε σε 2. Η 1η είναι η χρησιμοποίηση ενός **Embedding Layer** στο νευρωνικό μας δίκτυο και η 2η η εκτέλεση ενός ξεχωριστού **Doc2Vec** μοντέλου, με το οποίο μετατρέψαμε όλες τις περιλήψεις σε πολυδιάστατα διανύσματα και στη συνέχεια τις περάσαμε από ένα ξεχωριστό νευρωνικό δίκτυο για τις προβλέψεις των βαθμολογιών.

NN with Embedding Layer

Για την υλοποίηση με τον 1ο τρόπο, αρχικά υπολογίζουμε το μέγεθος του λεξικού των περιλήψεων των βιβλίων, καθώς και το μέγεθος της μεγαλύτερης περίληψης σε λέξεις με τη μέθοδο `calculateVocab`. Στη συνέχεια, εκπαιδεύουμε ένα νευρωνικό δίκτυο πάνω στις βαθμολογίες του cluster στον οποίο ανήκει ο χρήστης, μέσω της μεθόδου `trainClusterNetwork`.

```python
    # Use a single model with an embedding layer that vectorizes
    # summaries and later predicts missing ratings
    vocab_size, max_length = emb_layer_networks.calculateVocab(books)
    print(f"Vocab size: {vocab_size}, Max length: {max_length}")
    model = emb_layer_networks.trainClusterNetwork(users_cluster,
                                                   avg_clust_ratings, books,
                                                   vocab_size, max_length)
    combined_scores_clusters_nn, _ = functions.calculateCombinedScores(
        es_reply, user_id, use_cluster_ratings=True,
        avg_clust_ratings = avg_clust_ratings,
        cluster_assigned_users = cluster_assignement, use_nn=2,
        model=model, books=books, vocab_size=vocab_size, max_length=max_length)
    combined_scores_clusters_nn.to_csv(SCORES_W_CLUST_AND_NN_EMB, index=False)
```

main.py

Κατά την εκτέλεσή της, γίνεται σύζευξη των dataframes `avg_clust_ratings` και `books` ώστε να ανακτήσουμε τις περιλήψεις και τις μέσες βαθμολογίες των βιβλίων που έχουν βαθμολογηθεί από τον cluster του χρήστη. Στη συνέχεια, οι περιλήψεις περνάνε από επεξεργασία ώστε να μετατραπούν σε κατάλληλες εισόδους για το νευρωνικό μας με τη μέθοδο `getNetworkInput`, από όπου επιλέγουμε και το μοντέλο που θα χρησιμοποιηθεί. Τέλος, εκπαιδεύουμε το μοντέλο με τη μέθοδο `trainNetwork` και εμφανίζουμε γραφήματα σχετικά με την εκπαίδευσή του με τη μέθοδο `plotHistory`.

```python
def trainClusterNetwork(cluster: int, avg_clust_ratings: pd.DataFrame,
                        books: pd.DataFrame, vocab_size: int, max_length: int):
    """Train regression NN on user's cluster's average ratings."""

    print(f"Preparing network input for cluster {cluster}...")
    # Add book summaries to the avg_clust_ratings df
    print(avg_clust_ratings.loc[avg_clust_ratings["Cluster"]==cluster].head())
    cluster_ratings = pd.merge(
        right=avg_clust_ratings.loc[avg_clust_ratings["Cluster"]==cluster],
        left=books, on="isbn", validate="one_to_one")
    summaries = cluster_ratings["summary"].to_list()
    ratings = cluster_ratings["rating"].to_list()

    X, Y, model = getNetworkInput(summaries, ratings, vocab_size, max_length)
    print(f"Training network for cluster {cluster}...")

    # Print input and label shapes
    print(X.shape)
    print(Y.shape)
    # Print model summary
    model.summary()
    # Start training
    history = model.fit(X, Y, validation_split=0.3, epochs=10,
                        batch_size=10, verbose=1)
    plotHistory(history)

    return model
```

emb_layer_networks.py

Η μέθοδος getNetworkInput προ-επεξεργάζεται τις περιλήψεις με την preProcessSummaryv3, τις κωδικοποιεί με τη χρήση one hot encoding και τέλος προσθέτει μηδενικά (σαν κενές λέξεις ουσιαστικά) στις περιλήψεις που έχουν μικρότερο μέγεθος από τη μεγαλύτερη περίληψη (max_length). Τέλος, επιλέγεται το μοντέλο που θα χρησιμοποιηθεί στην εκπαίδευση, μετατρέπονται οι βαθμολογίες σε numpy array και επιστρέφεται η έτοιμη είσοδος του νευρωνικού.

```python
def getNetworkInput(summaries: list, ratings: list, vocab_size, max_length,
classifier = False):
    # Preprocess summaries
    preproc_summaries = [preProcessSummaryv3(summary) for summary in summaries]
    # One hot encode words of documents
    encoded_sums = [one_hot(d,vocab_size) for d in preproc_summaries]
    # Add padding
    X = pad_sequences(encoded_sums,maxlen=max_length,padding='post')
```

```
    # Normalize ratings
    if classifier:
        Y = np.array([customOHE2(r) for r in ratings])
        return (X, Y, classifier_model_1(vocab_size, max_length))
    # Define the model
    model = base_model_1(vocab_size, max_length)
    Y = np.array(ratings)
    return (X, Y, model)
```
emb_layer_networks.py

Η preProcessSummaryv3 μεταφράζει τους ανεπιθύμητους html escape χαρακτήρες (π.χ. "), τα σημεία στίξης και τα περιττά κενά και μετατρέπει όλους οι χαρακτήρες σε πεζοούς. Αυτό επιτυγχάνεται με τη χρήση μιας κανονικής έκφρασης και του html.unescape.

```
def preProcessSummaryv3(summary) -> str:
    return re.sub('[^A-Za-z0-9]+', ' ', html.unescape(summary))
```

Νευρωνικό δίκτυο

Το μοντέλο που επιλέχθηκε είναι ένα regression νευρωνικό δίκτυο. Αυτό σημαίνει ότι, σε αντίθεση με τα classification δίκτυα που κάνουν κατηγοριοποίηση, το μοντέλο μας έχει ως έξοδο μία τιμή. Το μοντέλο μας έχει ένα Embedding layer για την αναγνώριση των περιλήψεων, καθώς και 2 κρυφά layers με μεγέθη 256 και 64. Τέλος, το output layer αποτελείται από 1 νευρώνα, η τιμή του οποίου καθορίζει και την predicted τιμή του δικτύου για τη βαθμολογία ενός βιβλίου. Η συνάρτηση κόστους που επιλέχθηκε είναι το μέσο τετραγωνισμένο σφάλμα, καθώς θέλουμε να τιμωρούμε παραπάνω το δίκτυο για ακραίες τιμές σε σχέση με το μέσο απόλυτο σφάλμα, το οποίο όμως καταγράφουμε, αφού είναι πιο διαισθητική μετρική για να καταλάβουμε πόσο απέχουν κατά μέσο όρο οι προβλεπόμενες βαθμολογίες του δικτύου μας από τις πραγματικές.

```
def base_model_1(vocab_size, max_length):
    max_length = max_length
    # Create model
    model = Sequential()
    model.add(Embedding(input_dim=vocab_size, output_dim=128,
                        input_length=max_length))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1, activation='relu'))
    model.compile(optimizer='adam',loss="mse",metrics=['mae'])
    return model
```
emb_layer_networks.py

Τέλος, για την πρόβλεψη των τιμών τροποποιήσαμε τη μέθοδο `getAvgClusterRating` (και συνεπώς και τη `calculateCombinedScores` ως προς τα ορίσματά της) ώστε να μπορεί να χρησημοποιεί και το δίκτυο για την πρόβλεψη των τιμών που λείπουν, ανάλογα με τα ορίσματά της.

Ουσιαστικά αυτό που αλλάζει είναι ότι, αν δε βρει βαθμολογία ενός βιβλίου ούτε στο avg_clust_ratings, χρησιμοποιεί το μοντέλο για να προβλέψει τη βαθμολογία του βιβλίου. Πιο συγκεκριμένα, δέχεται ως όρισμα έναν ακέραιο use_nn ανάλογα με τη μέθοδο που χρησιμοποιούμε (Embedding layer ή Doc2Vec), καθώς και το μοντέλο, κάνει την απαραίτητη προεπεξεργασία της συνάρτησης και τέλος καλεί το model.predict() ώστε να υπολογίσει τη βαθμολογία.

```
def getAvgClusterRating(users_cluster: int, isbn: str, avg_clust_ratings:
                        pd.DataFrame, use_nn: int, model: Sequential = None,
                        vectorized_books: pd.DataFrame = None,
                        books: pd.DataFrame = None, vocab_size: int = None,
                        max_length: int = None) -> tuple:
    """Given a user id and an book's isbn, it returns the
    average rating of user's cluster for the specified book."""

    # Try getting user's cluster's rating of specified book
    try:
        rating = avg_clust_ratings.loc[(avg_clust_ratings["isbn"] == isbn) &
                                       (avg_clust_ratings["Cluster"] ==
                                        users_cluster)]["rating"].iloc[0]
        return rating
    # If a book hasn't been rated by a cluster and use_nn = 0,
    # return the median value of 5 stars out of 10
    except:
        # Using vectorized books
        if use_nn == 1:
            vect_sum = vectorized_books["Vectorized_Summary"].\
                loc[vectorized_books["isbn"] == isbn].to_list()[0]\
                    .reshape(1, -1)
            return model.predict(vect_sum)[0][0]
        # Not using vectorized books
        elif use_nn == 2:
            summary = books[books["isbn"]==isbn]["summary"].to_list()[0]
            # Preprocess summary
            preproc_summary = preProcessSummaryv3(summary)
            # One hot encode words of documents
            encoded_sum = one_hot(preproc_summary, vocab_size)
            # Add padding
            X = pad_sequences([encoded_sum],maxlen=max_length,padding='post')
```

```
            return model.predict(X)[0][0]

                                                                    functions.py
```

Παρατηρήσαμε ότι αυτή η υλοποίηση ήταν ελαφρώς καλύτερη από την 2η μας επιλογή, οπότε είναι και η κύρια που χρησιμοποιούμε. Η 2η μέθοδος δε θα αναλυθεί στην αναφορά αυτή, όμως βρίσκεται στον κώδικα σε σχόλια και δουλεύει κανονικά, αν και φαίνεται να είναι πιο αργή.

Έξοδος κατά την εκτέλεση

ElasticSearch

Τρέχουμε το ερώτημα “love” στην elastic ως ο χρήστης 11400 και παρατηρούμε τα 5 καλύτερα βιβλία. Όπως φαίνεται, τα περισσότερα βιβλία έχουν user_rating 5, που όμως δεν είναι η πραγματική βαθμολογία του χρήστη, έχει προκύψει επειδή την ορίσαμε εμείς ως default rating (αφού ο χρήστης δεν είχε βαθμολογήσει το αντίστοιχο βιβλίο). Παρόλα αυτά, το πρώτο βιβλίο το έχει βαθμολογήσει ο χρήστης με 8 και ενώ έχει σχετικά μικρό σκορ από την elastic (6.99/10), προωθείται πιο ψηλά στην κατάταξη λόγω της υψηλής βαθμολογίας του χρήστη.

```
Enter search string:love
Enter your user ID (must be an integer): 11400

Elasticsearch returned 9714 books. The 5 best matches are:
      score        isbn  ...                                            summary                    category
0  13.181017  0440153778  ...  Love poems describe a love affair, a painful b...                 ['poetry']
1  12.601347  0060925515  ...  With her trademark combination of candor, bras...                ['fiction']
2  12.569523  0679413936  ...  In Born For Love, Leo Buscaglia suggests that ...  ['family & relationships']
3  12.537282  0373872496  ...  The fourth title in the author&#39;s popular L...                ['fiction']
4  12.500157  037310443X  ...  First Love,Last Love by Carole Mortimer releas...         ['romance fiction']

[5 rows x 8 columns]

Calculating combined scores...
User 11400 has rated 62 book(s).

Best 5 matches without clustering:

       score  user_rating  es_scaled_score  ...      publisher                                            summary                    category
414  7.552128          8.0         6.992287  ...    Penguin USA  George Burns pays tribute to his partner, wife...  ['biography & autobiography']
0    7.222222          5.0        10.000000  ...           Dell  Love poems describe a love affair, a painful b...                 ['poetry']
1    7.026766          5.0         9.560224  ...  Harpercollins  With her trademark combination of candor, bras...                ['fiction']
2    7.016035          5.0         9.536080  ...          Slack  In Born For Love, Leo Buscaglia suggests that ...  ['family & relationships']
3    7.005164          5.0         9.511620  ...   Steeple Hill  The fourth title in the author&#39;s popular L...                ['fiction']

[5 rows x 10 columns]

Press enter to continue to Clustering...
```

Clustering

Επειδή ο χρήστης δεν υπάρχει στο αρχείο BX-Users.csv, δεν έχει νόημα να ξαναυπολογίσουμε τα σκορ των βιβλίων, αφού δε θα ανήκει σε κάποιον cluster.

```
Press enter to continue to Clustering...

Creating processed users CSV...
   User_ID  Age         Country  Country_ID
0        1   29             usa           0
1        2   18             usa           0
2        3   18          russia           1
3        4   17        portugal           2
4        5   87  united kingdom           3
Try loading pre-trained clustered users from file? (y/n): y
Loaded clustered users from file.
Re-calculating combined scores using user's cluster's average book ratings...
User 11400 has rated 62 book(s).
User 11400 doesn't exist in database.
Scores with clustering will be the same as scores without clustering.
```

Ξανατρέχουμε το πρόγραμμα απ την αρχή ως κάποιος χρήστης που αυτήν τη φορά υπάρχει στο BX-Users. Αυτή τη φορά επιλέξαμε το ερώτημα “animal”. Μετά από λίγη ώρα (αρκετή μάλλον λόγω εκτέλεσης σε λάπτοπ), προκύπτει η ομαδοποίηση των χρηστών σε συστάδες, σύμφωνα με την οποία ο χρήστης 12 ανήκει στη συστάδα .

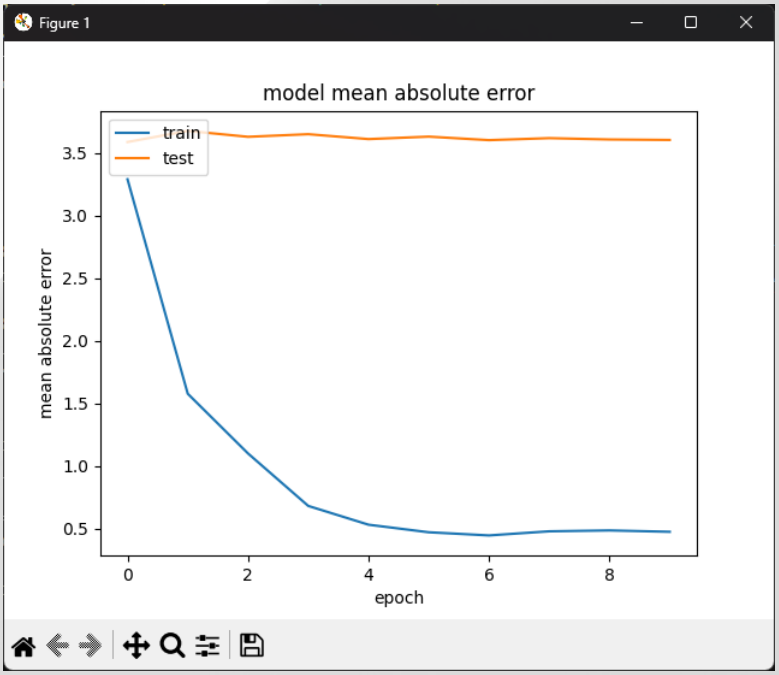
```
Enter search string:animal
Enter your user ID (must be an integer): 12

Elasticsearch returned 1754 books. The 5 best matches are:
       score        isbn  ...                                            summary             category
0  17.913660  0307101320  ...  Simple text and illustrations describe the hom...          ['animals']
1  17.662483  063113896X  ...  In Singer&#39;s book, the idea of the animal r...           ['nature']
2  17.576447  1570761914  ...  Babies and animals are a naturally adorable ma...  ['crafts & hobbies']
3  17.481062  0688081770  ...  Explains how the Sanctuary for Animals and the...     ['animal rescue']
4  17.358896  1930051220  ...  This is the true story of how the animal liber...           ['nature']

[5 rows x 8 columns]

Calculating combined scores...
User 12 has rated 1 book(s).

Best 5 matches without clustering:

      score  user_rating  es_scaled_score  ...                  publisher                                            summary             category
0  7.222222            5        10.000000  ...               Golden Books  Simple text and illustrations describe the hom...          ['animals']
1  7.159904            5         9.859785  ...         Blackwell Publishers  In Singer&#39;s book, the idea of the animal r...           ['nature']
2  7.138558            5         9.811757  ...  Trafalgar Square Publishing  Babies and animals are a naturally adorable ma...  ['crafts & hobbies']
3  7.114893            5         9.758509  ...          Morrow Junior Books  Explains how the Sanctuary for Animals and the...     ['animal rescue']
4  7.084583            5         9.690312  ...               Lantern Books  This is the true story of how the animal liber...           ['nature']

[5 rows x 10 columns]

Press enter to continue to Clustering...

Creating processed users CSV...
   User_ID  Age         Country  Country_ID
0        1   66             usa           0
1        2   18             usa           0
2        3   31          russia           1
3        4   17        portugal           2
4        5  120  united kingdom           3
Try loading pre-trained clustered users from file? (y/n): n
Clustering users...
Number of available threads: 8
Starting k-Prototypes using 8 threads...
Init: initializing centroids
Init: initializing centroids
Init: initializing centroids
Init: initializing centroids
Init: initializing centroids
Init: initializing centroids
Init: initializing centroids
Init: initializing centroids
```

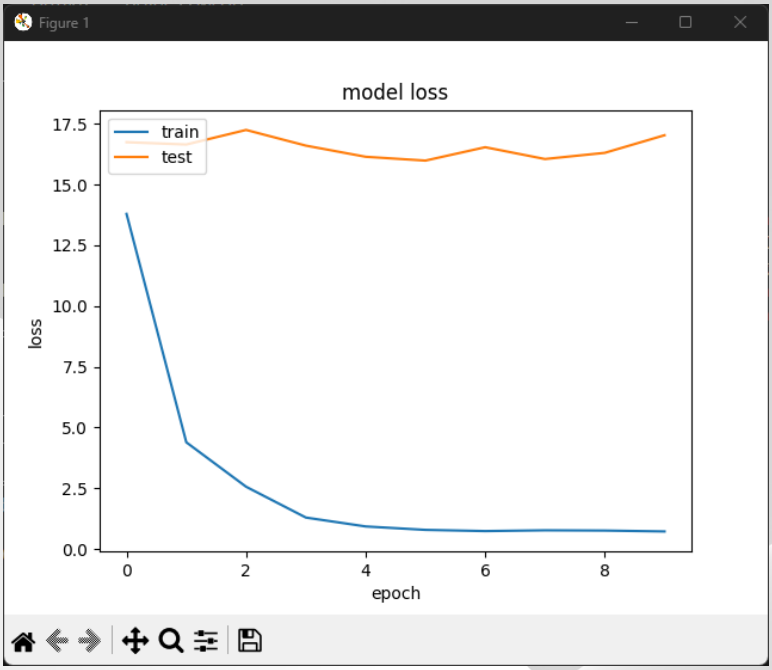
Συνυπολογίζοντας πλέον και τις μέσες βαθμολογίες της συστάδας, παρατηρούμε ότι συμπληρώνονται κάποιες τιμές από τις βαθμολογίες των ομάδων.

score	user_rating	es_scaled	isbn	book_title	book_author	year_of_publi	publisher	summary	category
7.464764	10	4.295719	789477645	Animal: The Definitive Vi	Don E. Wilson	2001	DK Publis	Offers	['nature']
7.447437	9	5.506733	767904311	Kindred Spirits : How the	ALLEN M. DVM MS SCH	2002	Broadway	A	['pets']
7.311985	5.833333333	9.160299	451526341	Animal Farm	George Orwell	2004	Signet	A satire	['fiction']
7.296898	6.5	8.293021	385314280	When Elephants Weep: Tl	Jeffrey Moussaieff Ma	1996	Delta	A study	['nature']
7.222222	5	10	307101320	Animals	Golden Books	1990	Golden Bc	Simple	['animals']
7.159904	5	9.859785	063113896X	In Defense of Animals	Peter Singer	1985	Blackwell	In	['nature']
7.138558	5	9.811757	1570761914	ANIMAL KNITS	Zoe Mellor	2001	Trafalgar S	Babies	['crafts & hobbies']
7.114893	5	9.758509	688081770	Safe in the spotlight: The	Elaine Scott	1991	Morrow Ju	Explains	['animal rescue']
7.084583	5	9.690312	1930051220	Free the Animals : The Stc	Ingrid Newkirk	2000	Lantern Bc	This is	['nature']
7.058979	5	9.632702	1884628192	Big Animals (Animal Stars	Paul Sterry	1995	Flying Fro	Briefly	['animals']
7.047435	5	9.606729	809245930	Balloon Animals	Aaron Hsu-Flanders	1988	McGraw-H	Theres	['balloon sculpture']
7.034279	5	9.577129	385242271	Animal Art	Lee J. Ames	1986	Bantam Dc	Step-by-st	['animals in art']
7.034279	5	9.577129	816735727	Animal ABC's	Susan Hood	1995	Troll Com	Animals fr	['juvenile nonfiction']
7.021553	10	3.298493	1592260012	Miffy at the Zoo (Miffy (Bi	Dick Bruna	2004	Big Tent E	Mr. Rabbit	['juvenile nonfiction']

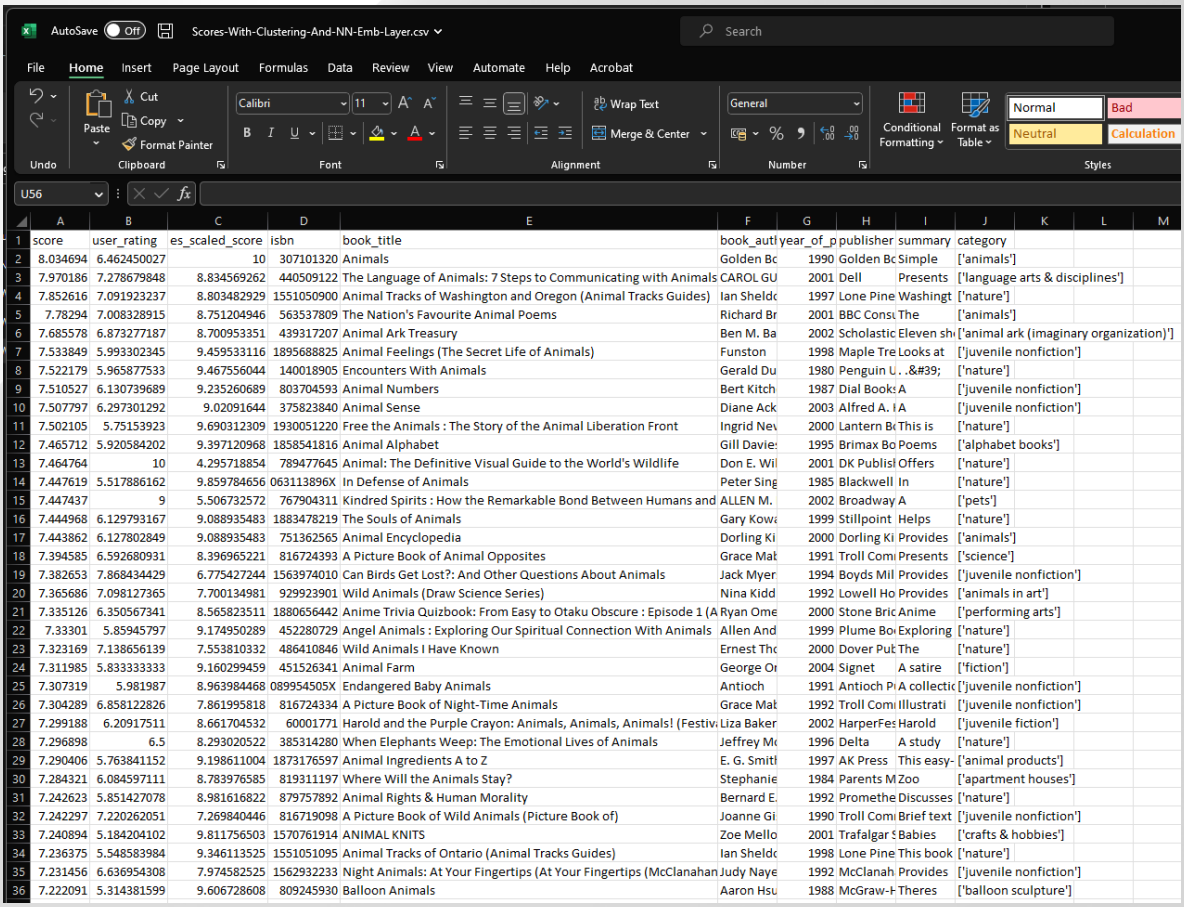
Νευρωνικό δίκτυο

Κατά την εκτέλεση του νευρωνικού δικτύου, παρατηρούμε ότι ενώ το κόστος αλλά και το μέσο απόλυτο σφάλμα ελαχιστοποιείται κατά την εκτέλεση, κατά την αξιολόγηση παραμένει σταθερό. Αυτό οφείλεται εν μέρει στα δεδομένα μας, καθώς δε φαίνεται να υπάρχει καθαρή συσχέτιση μεταξύ περιλήψεων και μέσων βαθμολογιών ανά ομάδα. Επομένως, αυτό που γίνεται είναι το μοντέλο να «αποστηθίζει» τις βαθμολογίες κατά την εκτέλεση και να μην ξέρει πώς να διαχειριστεί τις καινούριες περιλήψεις που δεν έχει δει. Αυτό το φαινόμενο λέγεται overfitting. Παρόλα αυτά, ακόμα και η μέση απόκλιση 3.5 είναι καλύτερη από τη μέση απόκλιση 5 που έχουμε όταν βάζουμε από μόνοι μας 5/10 στα βιβλία που δεν έχουν βαθμολογία.





Όπως παρατηρούμε, πλέον όλα τα βιβλία έχουν κάποια βαθμολογία διαφορετική του 5 στο user_rating, οι οποίες προκύπτουν κυρίως από τις μέσες βαθμολογίες συστάδων και το νευρωνικό.



score	user_rating	es_scaled_score	isbn	book_title	book_autl	year_of_p	publisher	summary	category
8.034694	6.462450027	10	307101320	Animals	Golden Bc	1990	Golden Bc	Simple	['animals']
7.970186	7.278679848	8.834569262	440509122	The Language of Animals: 7 Steps to Communicating with Animals	CAROL GU	2001	Dell	Presents	['language arts & disciplines']
7.852616	7.091923237	8.803482929	1551050900	Animal Tracks of Washington and Oregon (Animal Tracks Guides)	Ian Sheldc	1997	Lone Pine	Washingt	['nature']
7.78294	7.008328915	8.751204946	563537809	The Nation's Favourite Animal Poems	Richard Br	2001	BBC Const	The	['animals']
7.685578	6.873277187	8.700953351	439317207	Animal Ark Treasury	Ben M. Ba	2002	Scholastic	Eleven sh	['animal ark (imaginary organization)']
7.533849	5.993302345	9.459533116	1895688825	Animal Feelings (The Secret Life of Animals)	Funston	1998	Maple Tre	Looks at	['juvenile nonfiction']
7.522179	5.965877533	9.467556044	140018905	Encounters With Animals	Gerald Du	1980	Penguin U	. .'	['nature']
7.510527	6.130739689	9.235260689	803704593	Animal Numbers	Bert Kitch	1987	Dial Books	A	['juvenile nonfiction']
7.507797	6.297301292	9.02091644	375823840	Animal Sense	Diane Ack	2003	Alfred A. I	A	['juvenile nonfiction']
7.502105	5.75153923	9.690312309	1930051220	Free the Animals : The Story of the Animal Liberation Front	Ingrid Nev	2000	Lantern Bc	This is	['nature']
7.465712	5.920584202	9.397120968	1858541816	Animal Alphabet	Gill Davies	1995	Brimax Bo	Poems	['alphabet books']
7.464764	10	4.295718854	789477645	Animal: The Definitive Visual Guide to the World's Wildlife	Don E. Wil	2001	DK Publis	Offers	['nature']
7.447619	5.517886162	9.859784656	063113896X	In Defense of Animals	Peter Sing	1985	Blackwell	In	['nature']
7.447437	9	5.506732572	767904311	Kindred Spirits : How the Remarkable Bond Between Humans and	ALLEN M.	2002	Broadway	A	['pets']
7.444968	6.129793167	9.088935483	1883478219	The Souls of Animals	Gary Kowa	1999	Stillpoint	Helps	['nature']
7.443862	6.127802849	9.088935483	751362565	Animal Encyclopedia	Dorling Ki	2000	Dorling Ki	Provides	['animals']
7.394585	6.592680931	8.396965221	816724393	A Picture Book of Animal Opposites	Grace Mat	1991	Troll Com	Presents	['science']
7.382653	7.868434429	6.775427244	1563974010	Can Birds Get Lost?: And Other Questions About Animals	Jack Myer	1994	Boyds Mil	Provides	['juvenile nonfiction']
7.365686	7.098127365	7.700134981	929923901	Wild Animals (Draw Science Series)	Nina Kidd	1992	Lowell Ho	Provides	['animals in art']
7.335126	6.350567341	8.565823511	1880656442	Anime Trivia Quizbook: From Easy to Otaku Obscure : Episode 1 (A	Ryan Ome	2000	Stone Bric	Anime	['performing arts']
7.33301	5.85945797	9.174950289	452280729	Angel Animals : Exploring Our Spiritual Connection With Animals	Allen And	1999	Plume Bo	Exploring	['nature']
7.323169	7.138656139	7.553810332	486410846	Wild Animals I Have Known	Ernest Thc	2000	Dover Pub	The	['nature']
7.311985	5.833333333	9.160299459	451526341	Animal Farm	George Or	2004	Signet	A satire	['fiction']
7.307319	5.981987	8.963984468	089954505X	Endangered Baby Animals	Antioch	1991	Antioch P	A collectic	['juvenile nonfiction']
7.304289	6.858122826	7.861995818	816724334	A Picture Book of Night-Time Animals	Grace Mat	1992	Troll Com	Illustrati	['juvenile nonfiction']
7.299188	6.20917511	8.661704532	60001771	Harold and the Purple Crayon: Animals, Animals, Animals! (Festiv	Liza Baker	2002	HarperFes	Harold	['juvenile fiction']
7.296898	6.5	8.293020522	385314280	When Elephants Weep: The Emotional Lives of Animals	Jeffrey Mc	1996	Delta	A study	['nature']
7.290406	5.763841152	9.198611004	1873176597	Animal Ingredients A to Z	E. G. Smitl	1997	AK Press	This easy-	['animal products']
7.284321	6.084597111	8.783976585	819311197	Where Will the Animals Stay?	Stephanie	1984	Parents M	Zoo	['apartment houses']
7.242623	5.851427078	8.981616822	879757892	Animal Rights & Human Morality	Bernard E.	1992	Promethe	Discusses	['nature']
7.242297	7.220262051	7.269840446	816719098	A Picture Book of Wild Animals (Picture Book of)	Joanne Gi	1990	Troll Com	Brief text	['juvenile nonfiction']
7.240894	5.184204102	9.811756503	1570761914	ANIMAL KNITS	Zoe Mello	2001	Trafalgar S	Babies	['crafts & hobbies']
7.236375	5.548583984	9.346113525	1551051095	Animal Tracks of Ontario (Animal Tracks Guides)	Ian Sheldc	1998	Lone Pine	This book	['nature']
7.231456	6.636954308	7.974582525	1562932233	Night Animals: At Your Fingertips (At Your Fingertips (McClanahan	Judy Naye	1992	McClanah	Provides	['juvenile nonfiction']
7.222091	5.314381599	9.606728608	809245930	Balloon Animals	Aaron Hsu	1988	McGraw-H	Theres	['balloon sculpture']