Πανεπιστημιο Πειραια 2023

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ Τελικη Εργασια 2022-2023

Χρονόπουλος Κωνσταντίνος mppl21081

Εισαγωγη :

Μας έχει δοθεί ένα αρχείο “dataset.npy” που περιέχει στοιχεία για τοις βαθμολογίες που έκαναν οι χρήστες του imdb ,το κάθε entry είναι ένα string από μόνο του και τα στοιχεία είναι δομημένα κάπως έτσι:

userID, movieID, rating, review date

Παραδείγματος χάρι :

ur18238764, tt2177461, 9, 22 January 2019

Άρα στο συγκεκριμένο παράδειγμα ο χρήστης με κωδικό “ur18238764” βαθμολόγησε την ταινία με κωδικό “tt2177461” με 9 στης “22 January 2019”.

Όλοι οι χρήστες έχουν κωδικό που είναι της μορφής ur********* και όλες οι ταινίες έχουν κωδικό που είναι της μορφής tt******** .

New dataset :

Επειδή το αρχείο “dataset,npy” είναι πάρα πολύ μεγάλο πάνω από 700mb και 4 εκατομμύρια entry’s δημιουργήσαμε ένα νέο ”newdataset.npy” που έχει 10000 τυχαία entry’s από το αρχικό για να δοκιμάζουμε τον κώδικα μας πιο εύκολα.

Εξήγηση του κώδικα :

```
# Load the npy file
dataset = np.load('dataset.npy', allow_pickle=True)

# Load the data using numpy.genfromtxt()
df = pd.DataFrame(np.genfromtxt(dataset, delimiter=',', dtype='str'),
                  columns=["userID", "movieID", "rating",
"review_date"])

# Define the minimum and maximum number of votes
Rmin = 3
Rmax = 10

# Group the data by "userID" and count the number of occurrences
user_vote_counts = df.groupby("userID").size()

# Filter the "userID" that satisfy the condition Rmin <= count <=
Rmax
valid_users = user_vote_counts[(user_vote_counts >= Rmin) &
(user_vote_counts <= Rmax)].index

# Filter the DataFrame to include only the valid users
filtered_df = df[df["userID"].isin(valid_users)]

# Group the data by "userID" and count the number of occurrences
user_vote_counts = df.groupby("userID").size()

# Use value_counts() to get the count of users for each vote count
user_count_by_vote = user_vote_counts.value_counts()

# Group the data by "userID" and count the number of occurrences
user_vote_counts = df.groupby("userID").size()

# Convert "review_date" column to datetime for calculating temporal
range
df['review_date'] = pd.to_datetime(df['review_date'])

# Calculate the temporal range (in days) for each user
user_temporal_range = (df.groupby("userID")['review_date'].max() -
df.groupby("userID")['review_date'].min()).dt.days

# Create dictionaries to map "userID" and "movieID" to integer
indices
user_id_map = {user_id: i for i, user_id in
enumerate(df["userID"].unique())}
movie_id_map = {movie_id: i for i, movie_id in
enumerate(df["movieID"].unique())}

algorithm = ""
```

Σε αυτό το απόσπασμα κώδικα φορτώνουμε το αρχείο numpy με όνομα "dataset.npy", το οποίο περιέχει τις αξιολογήσεις χρηστών για τοις ταινίες, και φιλτράρουμε το σύνολο δεδομένων για να περιλαμβάνει μόνο τους χρήστες που έχουν ψηφίσει σε ένα συγκεκριμένο εύρος ταινιών, με αριθμούς ψήφων μεταξύ Rmin και Rmax. Στη συνέχεια ομαδοποιούμε το φιλτραρισμένο σύνολο δεδομένων ανά "userID" και μετράμε τον αριθμό εμφανίσεων για κάθε χρήστη, αποθηκεύουμε τους μετρητές ψήφων των χρηστών, και

παίρνουμε τον αριθμό των χρηστών για κάθε μετρητή ψήφων. Επίσης, υπολογίζουμε το χρονικό εύρος για κάθε χρήστη, το οποίο είναι ο αριθμός των ημερών μεταξύ της πρώτης και της τελευταίας ημερομηνίας αξιολόγησης. Τέλος, δημιουργούμε λεξικά για να αντιστοιχίζει το "userID" και το "movieID" σε ακέραιους δείκτες και αρχικοποιούμε την μεταβλητή 'algorithm' σε μια κενή συμβολοσειρά.

Πιο συγκεκριμένα, πραγματοποιεί τις εξής ενέργειες:

1. Φορτώνουμε το αρχείο numpy με το όνομα "dataset.npy" χρησιμοποιώντας την συνάρτηση np.load() και το μετατρέπουμε σε ένα DataFrame του pandas χρησιμοποιώντας την συνάρτηση pd.DataFrame().
2. Ορίζουμε το ελάχιστο και μέγιστο αριθμό ψήφων, Rmin και Rmax, που πρέπει να έχει ένας χρήστης για να περιλαμβάνεται στο φιλτραρισμένο σύνολο δεδομένων.
3. Ομαδοποιούμε το φιλτραρισμένο DataFrame με βάση το "userID" και μετράμε τον αριθμό εμφανίσεων για κάθε χρήστη χρησιμοποιώντας τις συναρτήσεις .groupby() και .size().
4. Φιλτράρουμε τα "userID" που ικανοποιούν τη συνθήκη Rmin <= count <= Rmax χρησιμοποιώντας ευρετηρίαση(indexing) και αποθηκεύουμε τους έγκυρους χρήστες σε μια μεταβλητή με το όνομα 'valid_users'.
5. Φιλτράρουμε το DataFrame ώστε να περιλαμβάνει μόνο τους έγκυρους χρήστες χρησιμοποιώντας τη μέθοδο .isin() και boolean indexing και αποθηκεύουμε το φιλτραρισμένο DataFrame ως 'filtered_df'
6. Ομαδοποιούμε το φιλτραρισμένο DataFrame ανά "userID" και μετράμε τον αριθμό εμφανίσεων για κάθε χρήστη χρησιμοποιώντας τη μέθοδο .groupby() και .size() ξανά.
7. Χρησιμοποιούμε τη μέθοδο .value_counts() για να πάρουμε τον αριθμό των χρηστών για κάθε αριθμό ψήφων.
8. Μετατρέπουμε τη στήλη "review_date" σε μορφή ημερομηνίας χρησιμοποιώντας την pd.to_datetime().
9. Υπολογίζουμε την χρονική διάρκεια (σε ημέρες) για κάθε χρήστη αφαιρώντας την πρώτη ημερομηνία κριτικής από την τελευταία ημερομηνία κριτικής χρησιμοποιώντας .groupby(), .max(), .min(), και .dt.days.
10. Δημιουργούμε λεξικά για να αντιστοιχίζουμε "userID" και "movieID" σε ακέραιους δείκτες χρησιμοποιώντας λεξικογραφική συμπτυγματολογία(dictionary comprehension).
11. Αρχικοποιούμε τη μεταβλητή 'algorithm' σε ένα κενό string.

Μετέπιπτα έχουμε :

```
while True:
    # Display menu options
    print("Select an option:")
    print("1. Print number of entries, unique users, and unique
movies")
    print("2. Print filtered DataFrame for valid users")
    print("3. Print user count by vote count")
    print("4. Plot histogram of number of votes per user")
    print("5. Plot histogram of temporal range of reviews per user")
    print("6. Create an alternative representation of the dataset as
preference vectors")
    print("7. Cluster the limited set of users into L clusters using
euclidian")
    print("8. Cluster the limited set of users into L clusters
cosine")
    print("9. Visualize the clusters")
    print("10. Cluster the limited set of users into L clusters ")
    print("11.create a collaborative filtering recommendation system
using a neural network")
    print("0. Exit")
    # Get user input
    choice = input("Enter your choice (1-11): ")
```

Σε αυτό το τμήμα κώδικα εμφανίζουμε ένα μενού επιλογών για τον χρήστη, και ζητάμε από τον χρήστη να εισάγει μια επιλογή μεταξύ 1 και 11, ή 0 για έξοδο. Χρησιμοποιούμε ένα while loop για να εμφανίζει συνεχώς το μενού και να παίρνει είσοδο από τον χρήστη μέχρι ο χρήστης να επιλέξει να βγει εισάγοντας 0.

Πιο συγκεκριμένα, ο κώδικας περιλαμβάνει τα ακόλουθα βήματα:

Ξεκινάει έναν άπειρο βρόχο χρησιμοποιώντας το while True.

Εμφανίζει ένα μενού επιλογών χρησιμοποιώντας εντολές print().

Ζητάει από τον χρήστη να εισάγει μια επιλογή χρησιμοποιώντας το input().

Αποθηκεύει την επιλογή του χρήστη σε μια μεταβλητή με το όνομα 'choice'.

Ο βρόχος συνεχίζεται μέχρι ο χρήστης να εισάγει το 0 για έξοδο από το πρόγραμμα.

Επιλογή 1:

```
if choice == '1':
    print("Number of entries: ", len(df))
    print("Number of unique users: ", len(df["userID"].unique()))
    print("Number of unique movies: ", len(df["movieID"].unique()))
```

Σε αυτό το κομμάτι κώδικα εκτυπώνουμε τον αριθμό των εγγραφών, μοναδικών χρηστών και μοναδικών ταινιών στον δεδομένο πίνακα δεδομένων 'df'.

Πιο συγκεκριμένα, πραγματοποιεί τις εξής ενέργειες:

Εκτυπώνει το αλφαριθμητικό " Number of entries: ", ακολουθούμενο από το μήκος του πίνακα δεδομένων df χρησιμοποιώντας τη συνάρτηση len (). Αυτό παρέχει τον συνολικό αριθμό των γραμμών στον πίνακα δεδομένων.

Εκτυπώνει το αλφαριθμητικό " Number of unique users: ", ακολουθούμενο από το μήκος των μοναδικών userIDs στο DataFrame. Αυτό δίνει τον αριθμό των διακριτών χρηστών που έχουν αξιολογήσει ταινίες στο σύνολο δεδομένων.

Εκτυπώνει το αλφαριθμητικό " Number of unique movies: ", ακολουθούμενο από το μήκος των μοναδικών movieIDs στο DataFrame. Αυτό δίνει τον αριθμό των διακριτών ταινιών που έχουν αξιολογηθεί στο σύνολο δεδομένων.

Επιλογή 2:

```
elif choice == '2':
    print(filtered_df)
```

Σε αυτό το τμήμα κώδικα εκτυπώνουμε το φιλτραρισμένο DataFrame 'filtered_df', το οποίο περιέχει μόνο τους έγκυρους χρήστες που έχουν βαθμολογήσει ανάμεσα σε Rmin και Rmax ταινίες.

Συγκεκριμένα, πραγματοποιεί την ακόλουθη ενέργεια:

Εκτυπώνει ολόκληρο το DataFrame 'filtered_df', το οποίο περιλαμβάνει μόνο τις σειρές όπου η στήλη "userID" είναι στο σύνολο των έγκυρων χρηστών που έχουν ανάμεσα σε Rmin και Rmax βαθμολογίες.

Επιλογή 3:

```
elif choice == '3':
    print(user_count_by_vote)
```

Σε αυτό το τμήμα κώδικα εκτυπώνουμε τον αριθμό των χρηστών για κάθε αριθμό ψήφων στο DataFrame "df".

Συγκεκριμένα, πραγματοποιεί την ακόλουθη ενέργεια:

Υπολογίζουμε τον αριθμό των χρηστών για κάθε αριθμό ψήφων χρησιμοποιώντας τη μέθοδο 'value_counts ()' στη pandas series.'user_vote_counts'.

Εκτυπώνει την παραγόμενη pandas series 'user_count_by_vote', η οποία περιέχει τον αριθμό των χρηστών που έχουν βαθμολογήσει ένα συγκεκριμένο αριθμό ταινιών.

Επιλογή 4:

```
elif choice == '4':
    plt.figure(figsize=(8, 6))
    plt.hist(user_vote_counts, bins=20, color='skyblue')
    plt.xlabel("Number of Votes")
    plt.ylabel("Frequency")
    plt.title("Histogram of Number of Votes per User")
    plt.show()
```

Σε αυτό το απόσπασμα κώδικα δημιουργουμε ένα ιστόγραμμα του αριθμού των ψήφων ανά χρήστη στο DataFrame "df" χρησιμοποιώντας τη βιβλιοθήκη matplotlib.

Πιο συγκεκριμένα, πραγματοποιεί την ακόλουθη ενέργεια:

Δημιουργεί ένα νέο σχήμα matplotlib με μέγεθος 8x6 ίντσες χρησιμοποιώντας το 'plt.figure()'.

Σχεδιάζει ένα ιστόγραμμα του αριθμού των ψήφων ανά χρήστη χρησιμοποιώντας το 'plt.hist()', όπου τα δεδομένα είναι μια pandas series "user_vote_counts", ο αριθμός των κάδων (bins) έχει οριστεί σε 20 και ο χρωματισμός έχει οριστεί σε "skyblue".

Ορίζει την ετικέτα x ως "Number of Votes" χρησιμοποιώντας το 'plt.xlabel()'.

Ορίζει την ετικέτα y ως "Frequency" χρησιμοποιώντας το 'plt.ylabel()'.

Ορίζει τον τίτλο του γραφήματος ως "Histogram of Number of Votes per User" χρησιμοποιώντας το 'plt.title()'.

Εμφανίζει το γράφημα χρησιμοποιώντας το 'plt.show()'.

Επιλογή 5:

```
elif choice == '5':
    plt.figure(figsize=(8, 6))
    plt.hist(user_temporal_range, bins=20, color='lightgreen')
    plt.xlabel("Temporal Range (days)")
    plt.ylabel("Frequency")
    plt.title("Histogram of Temporal Range of Reviews per User")
    plt.show()
```

Σε αυτό το τμήμα κώδικα δημιουργούμε ένα ιστόγραμμα του χρονικού εύρους (σε ημέρες) των κριτικών ανά χρήστη στο DataFrame 'df' χρησιμοποιώντας τη βιβλιοθήκη matplotlib.

Πιο συγκεκριμένα, πραγματοποιουμε την ακόλουθη ενέργεια:

Δημιουργούμε ένα νέο σχήμα matplotlib με μέγεθος 8x6 ιντσών χρησιμοποιώντας την εντολή 'plt.figure()'.

Σχεδιάζουμε ένα ιστόγραμμα του χρονικού εύρους των κριτικών ανά χρήστη χρησιμοποιώντας την εντολή 'plt.hist()', όπου τα δεδομένα είναι η pandas series 'user_temporal_range', ο αριθμός των κάδων (bins) ορίζεται σε 20 και το χρώμα ορίζεται σε 'lightgreen'.

Ορίζουμε τηνετικέτα x ως " Temporal Range (days) " χρησιμοποιώντας την εντολή 'plt.xlabel()'.

Ορίζουμε την ετικέτα y ως " Frequency " χρησιμοποιώντας την εντολή 'plt.ylabel()'.

Ορίζουμε τον τίτλο του γραφήματος ως " "Histogram of Temporal Range of Reviews per User " χρησιμοποιώντας την εντολή 'plt.title()'.

Επιδικνιουμε το γράφημα χρησιμοποιώντας την εντολή 'plt.show()'.

Επιλογή 6:

```
elif choice == '6':
    # Create an alternative representation of the dataset as
preference vectors
    num_users = len(user_id_map)
    num_movies = len(movie_id_map)

    # Create a sparse matrix to represent the preference vectors
    preference_matrix = lil_matrix((num_users, num_movies),
dtype=np.uint8)

    # Group the filtered DataFrame by user ID
    grouped_df = filtered_df.groupby("userID")

    # Loop through each group and populate the preference matrix
    for user_id, group in grouped_df:
        user_idx = user_id_map[user_id]
        for _, row in group.iterrows():
            movie_id = row["movieID"]
            rating = row["rating"]
            movie_idx = movie_id_map[movie_id]
            preference_matrix[user_idx, movie_idx] = rating
    print(preference_matrix)
```

Σε αυτό το τμήμα κώδικα δημιουργούμε μια εναλλακτική αναπαράσταση του συνόλου δεδομένων ως διανυσματικών προτιμήσεων. Αρχικά υπολογίζει τον αριθμό των χρηστών και ταινιών και στη συνέχεια δημιουργεί έναν πίνακα σχήματος (num_users, num_movies) για να αναπαραστήσει τα διανυσματικά προτιμήσεων.

Στη συνέχεια ομαδοποιουμε το DataFrame που φιλτράρεται ανά ID χρήστη και κάνοθμε βρόχο σε κάθε ομάδα για να γεμίσουμε τον πίνακα προτιμήσεων. Για κάθε γραμμή στην ομάδα, εξάγουμε το ID της ταινίας, τη βαθμολογία και το ID του χρήστη και τα αντιστοιχίζουμε στους αντίστοιχους δείκτες στον πίνακα προτιμήσεων χρησιμοποιώντας τα λεξικά user_id_map και movie_id_map. Στη συνέχεια θέτουμε την τιμή στο αντίστοιχο κελί του πίνακα προτιμήσεων στη βαθμολογία.

Τέλος, εκτυπώνουμε τον πίνακα προτιμήσεων.

Επιλογή 7:

```
elif choice == '7':
    algorithm = "eucl"
    # Perform k-means clustering on the preference matrix
    num_clusters = 10  # choose the number of clusters
    kmeans = KMeans(n_clusters=num_clusters)
    cluster_labels = kmeans.fit_predict(preference_matrix)

    cluster_counts = Counter(cluster_labels)
    for label, count in cluster_counts.items():
        print(f"Cluster {label} has {count} members.")
    print(algorithm)
```

Σε αυτό το τμήμα κώδικα εκτελούμε k-means clustering στο preference matrix που δημιουργήθηκε στο προηγούμενο βήμα (βήμα 6) και εκτυπώνουμε τον αριθμό των μελών σε κάθε cluster.

Η μεταβλητή num_clusters έχει οριστεί σε 10, που σημαίνει ότι ο αλγόριθμος k-means θα ομαδοποιήσει τους χρήστες σε 10 clusters βάσει των προτιμήσεών τους.

Χρησιμοποιήσουμε την κλάση KMeans από το module sklearn.cluster για την εκτέλεση του clustering. Η μέθοδος fit_predict() καλείται για να ταιριάξει το μοντέλο στο preference matrix και να προβλέψει τις ετικέτες των cluster για κάθε χρήστη.

Χρησιμοποιούμε την κλάση Counter από το module collections για να μετρήσουμε τον αριθμό των χρηστών σε κάθε cluster, και τα αποτελέσματα εκτυπώνονται χρησιμοποιώντας ένα for loop.

Τέλος, ορίζουμε την μεταβλητή algorithm ως "eucl".

Επιλογή 8:

```
elif choice == '8':
    algorithm = "cosin"
    # Create the user-item matrix
    n_users = len(user_id_map)
    n_items = len(movie_id_map)
    user_indices = np.array([user_id_map[user_id] for user_id in
filtered_df['userID']])
    item_indices = np.array([movie_id_map[movie_id] for movie_id in
filtered_df['movieID']])
    ratings = np.array(filtered_df['rating'], dtype=np.float32)
    user_item_matrix = coo_matrix((ratings, (user_indices,
item_indices)), shape=(n_users, n_items)).tocsr()

    # Compute the cosine similarity matrix
    cosine_sim_matrix = cosine_similarity(user_item_matrix,
dense_output=False)

    # Use MiniBatchKMeans to cluster the users based on their ratings
    n_clusters = 5  # Set the number of clusters to 5
    kmeans = MiniBatchKMeans(n_clusters=n_clusters, random_state=42)
    user_cluster_labels = kmeans.fit_predict(cosine_sim_matrix)

    # Group the users into clusters
    user_clusters = [set() for _ in range(n_clusters)]
    for user_idx, cluster_label in enumerate(user_cluster_labels):
        user_clusters[cluster_label].add(user_idx)

    # Print the number of users in each cluster
    for i in range(n_clusters):
        print(f"Cluster {i + 1}: {len(user_clusters[i])} users")
```

Σε αυτό το κομμάτι του κώδικα εκτελούμε ομαδοποίηση χρηστών με βάση τις αξιολογήσεις ταινιών τους χρησιμοποιώντας τον πίνακα ομοιότητας συνημίτονου και τον αλγόριθμο MiniBatchKMeans από τη βιβλιοθήκη scikit-learn.

Στον κώδικα δημιουργούμε πρώτα μια μήτρα στοιχείου χρήστη με βάση το DataFrame 'filtered_df', η οποία περιέχει πληροφορίες σχετικά με τους χρήστες, τις ταινίες και τις αντίστοιχες αξιολογήσεις τους. Αυτός ο πίνακας είναι ένας αραιός πίνακας, ο οποίος αναπαρίσταται σε μορφή COO χρησιμοποιώντας τη συνάρτηση 'coo_matrix()'.

Στη συνέχεια, ο πίνακας ομοιότητας συνημίτονου υπολογίζεται από τον πίνακα στοιχείων χρήστη χρησιμοποιώντας τη συνάρτηση 'cosine_similarity()' από τη βιβλιοθήκη scikit-learn. Αυτός ο πίνακας περιέχει τις βαθμολογίες ομοιότητας κατά ζεύγη συνημίτονου μεταξύ κάθε ζεύγους χρηστών στο σύνολο δεδομένων.

Στη συνέχεια, χρησιμοποιούμε τον αλγόριθμος MiniBatchKMeans για την ομαδοποίηση των χρηστών σε «n_clusters» αριθμό συμπλεγμάτων. Ο αλγόριθμος αντιστοιχίζει σε κάθε χρήστη ένα σύμπλεγμα με βάση τις βαθμολογίες ομοιότητας συνημίτονου τους, οι οποίες αντιπροσωπεύουν την ομοιότητά τους όσον αφορά τις προτιμήσεις ταινιών τους. Η μεταβλητή 'user_cluster_labels' αποθηκεύει την ετικέτα συμπλέγματος που έχει εκχωρηθεί σε κάθε χρήστη.

Στη συνέχεια, ομαδοποιούμε τους χρήστες σε συμπλέγματα με βάση τις ετικέτες συμπλέγματος που τους έχουν εκχωρηθεί. Η μεταβλητή 'user_clusters' είναι μια λίστα συνόλων, όπου κάθε σύνολο περιέχει τους δείκτες των χρηστών στο αντίστοιχο σύμπλεγμα.

Τέλος, εκτυπώνουμε τον αριθμό των χρηστών σε κάθε σύμπλεγμα χρησιμοποιώντας έναν βρόχο for που επαναλαμβάνεται στον αριθμό των συμπλεγμάτων.

Επιλογή 9:

```
elif choice == '9':
    print(algorithm)
    if algorithm == "eucl":
        # Apply TruncatedSVD to reduce the dimensionality of the
preference matrix
        svd = TruncatedSVD(n_components=2)
        svd_matrix = svd.fit_transform(preference_matrix)

        # Plot the clusters using the TruncatedSVD-reduced matrix
        plt.scatter(svd_matrix[:, 0], svd_matrix[:, 1],
c=cluster_labels)
        plt.title('K-means Clustering of Users')
        plt.xlabel('TruncatedSVD Dimension 1')
        plt.ylabel('TruncatedSVD Dimension 2')
        plt.show()
    elif algorithm == "cosin":
        # Visualize the clusters
        pca = PCA(n_components=2)
        svd = TruncatedSVD(n_components=2)
        cosine_sim_matrix_reduced =
svd.fit_transform(cosine_sim_matrix)

        plt.figure(figsize=(10, 10))
        for i in range(n_clusters):
            cluster_data =
cosine_sim_matrix_reduced[list(user_clusters[i]), :]
            plt.scatter(cluster_data[:, 0], cluster_data[:, 1],
label=f"Cluster {i + 1}")
        plt.legend()
        plt.show()
    else:
        print("You have not created the clusters yet")
```

Σε αυτό το απόσπασμα του κώδικα οπτικοποιούμε τα συμπλέγματα χρηστών που δημιουργήθηκαν στο προηγούμενο απόσπασμα κώδικα, χρησιμοποιώντας είτε την ευκλείδεια απόσταση είτε το μέτρο ομοιότητας συνημίτονου.

Ο κώδικας ελέγχει πρώτα την τιμή της μεταβλητής 'algorithm', η οποία είναι μια συμβολοσειρά που καθορίζει το μέτρο απόστασης που χρησιμοποιείται για την ομαδοποίηση. Εάν είναι 'eucl', ο κώδικας εφαρμόζει τον αλγόριθμο TruncatedSVD για να μειώσει τη διάσταση του πίνακα προτιμήσεων σε δύο διαστάσεις. Η συνάρτηση 'svd.fit_transform()' χρησιμοποιείται για την εκτέλεση αυτού του μετασχηματισμού. Ο μειωμένος πίνακας στη συνέχεια σχεδιάζεται χρησιμοποιώντας το 'plt.scatter()', όπου οι συντεταγμένες x και y είναι οι δύο διαστάσεις που λαμβάνονται από τον αλγόριθμο TruncatedSVD και το χρώμα κάθε σημείου αντιπροσωπεύει την ετικέτα συμπλέγματος του αντίστοιχου χρήστη. Στη συνέχεια, η γραφική παράσταση εμφανίζεται χρησιμοποιώντας το 'plt.show()'.

Εάν η μεταβλητή "algorithm" είναι "cosin", ο κώδικας οπτικοποιούμε τα συμπλέγματα χρησιμοποιώντας τους αλγόριθμους PCA και TruncatedSVD για να μειώσει τη διάσταση του πίνακα ομοιότητας συνημιτόνου σε δύο διαστάσεις. Η συνάρτηση 'svd.fit_transform()' χρησιμοποιείται για την εκτέλεση του μετασχηματισμού TruncatedSVD. Ο μειωμένος πίνακας σχεδιάζεται χρησιμοποιώντας το 'plt.scatter()', όπου κάθε σύμπλεγμα απεικονίζεται με διαφορετικό χρώμα και τα σημεία σε κάθε σύμπλεγμα λαμβάνονται με

ευρετηρίαση στον πίνακα 'cosine_sim_matrix_reduced' χρησιμοποιώντας τη λίστα 'user_clusters'. Στη συνέχεια, η γραφική παράσταση εμφανίζεται χρησιμοποιώντας το 'plt.show()'.

Εάν η μεταβλητή «αλγόριθμος» δεν είναι ούτε «eucl» ούτε «cosin», απλώς εκτυπώνουμε ένα μήνυμα που υποδεικνύει ότι τα συμπλέγματα δεν έχουν δημιουργηθεί ακόμη.

Επιλογή 10:

```
elif choice == '10':
    # Perform PCA to reduce the dimensionality of the preference
matrix
    pca = PCA(n_components=50)
    preference_matrix_pca =
pca.fit_transform(preference_matrix.toarray())
    # Compute the pairwise Jaccard distance between the rows of the
reduced preference matrix
    jaccard_dist = pdist(preference_matrix_pca, metric='jaccard')
    # Convert the pairwise distance vector to a distance matrix
    jaccard_dist_matrix = squareform(jaccard_dist)
    print(jaccard_dist_matrix)

    # Compute the hierarchical clustering of the Jaccard distance
matrix
    linkage_matrix = linkage(jaccard_dist_matrix, method='ward')

    # Plot the dendrogram of the hierarchical clustering
    #         plt.figure(figsize=(10, 5))
    #         dendrogram(linkage_matrix, truncate_mode='level', p=10)
    #         plt.xlabel('Users')
    #         plt.ylabel('Jaccard distance')
    #         plt.show()

    # Assign each user to a cluster based on the hierarchical
clustering
    num_clusters = 3
    user_clusters = fcluster(linkage_matrix, num_clusters,
criterion='maxclust')
```

Σε αυτό το απόσπασμα του κώδικα εκτελουμε ιεραρχική ομαδοποίηση σε μια μήτρα στοιχείου χρήστη χρησιμοποιώντας την απόσταση Jaccard ανά ζεύγος μεταξύ των σειρών του πίνακα.

Πιο συγκεκριμένα, εκτελουμε τις ακόλουθες ενέργειες:

Μειώνουμε τη διάσταση του πίνακα προτιμήσεων χρησιμοποιώντας PCA σε 50 στοιχεία.

Υπολογίζουμε την κατά ζεύγη απόσταση Jaccard μεταξύ των σειρών του πίνακα μειωμένων προτιμήσεων χρησιμοποιώντας τη συνάρτηση pdist από τη βιβλιοθήκη scipy.

Μετατρέππυμε το διάνυσμα απόστασης κατά ζεύγη σε πίνακα απόστασης χρησιμοποιώντας τη συνάρτηση τετραγωνικής μορφής από τη βιβλιοθήκη scipy.

Υπολογίζουμε την ιεραρχική ομαδοποίηση του πίνακα απόστασης Jaccard χρησιμοποιώντας τη συνάρτηση σύνδεσης από τη βιβλιοθήκη scipy με τη μέθοδο 'ward'.

Αντιστοιχίζουμε σε κάθε χρήστη ένα σύμπλεγμα με βάση την ιεραρχική ομαδοποίηση χρησιμοποιώντας τη συνάρτηση fcluster από τη βιβλιοθήκη scipy με το κριτήριο «maxclust».

Επιλογή 11:

```
elif choice == '11':
    L = 5
    predictions = {}

    for a in range(L):
        cluster_users = np.where(user_clusters == a)[0]

        if len(cluster_users) == 0:
            continue  # skip this cluster if there are no users

        # Extract the subset of the preference matrix that
corresponds to the users in the cluster
        cluster_preferences = preference_matrix[cluster_users, :]


        cosine_sim = cosine_similarity(cluster_preferences)

        k = 5
        closest_neighbors = np.argsort(-cosine_sim, axis=1)[:, 1:k +
1]

        # Define the neural network architecture
        inputs = layers.Input(shape=(k,))
        x = layers.Dense(64, activation='relu')(inputs)
        x = layers.Dropout(0.2)(x)
        x = layers.Dense(32, activation='relu')(x)
        x = layers.Dropout(0.2)(x)
        outputs = layers.Dense(1, activation='linear')(x)

        model = keras.Model(inputs=inputs, outputs=outputs)

        # Compile the model
        model.compile(optimizer='adam', loss='mse')

        # Prepare the training data
        X_train = np.zeros((len(cluster_users), k))
        y_train = np.zeros((len(cluster_users), 1))

        for i, user in enumerate(cluster_users):
            X_train[i, :] = preference_matrix[closest_neighbors[i],
:].toarray().flatten()[:k]
            y_train[i, 0] = preference_matrix[user, 0]

        # Train the neural network
        model.fit(X_train, y_train, epochs=10, batch_size=32)

        # Use the trained neural network to predict the ratings of
each user in the cluster
        X_pred = np.zeros((len(cluster_users), k))

        for i, user in enumerate(cluster_users):
            X_pred[i, :] = preference_matrix[closest_neighbors[i],
:].toarray().flatten()[:5]

        y_pred = model.predict(X_pred)

        # Store the predicted ratings in a dictionary
        predictions[a] = (cluster_users, y_pred)
```

Σε αυτό το απόσπασμα του κώδικα υλοποιουμε ένα σύστημα συστάσεων χρησιμοποιώντας ένα νευρωνικό δίκτυο. Πρώτα εκτελουμε ομαδοποίηση στους χρήστες με βάση τις προτιμήσεις τους και, στη συνέχεια, για κάθε σύμπλεγμα, εκπαιδεύουμε ένα νευρωνικό δίκτυο για να προβλέψει τις αξιολογήσεις των χρηστών με βάση τις αξιολογήσεις των πλησιέστερων γειτόνων τους. Στη συνέχεια, αποθηκεύουμε τις προβλεπόμενες βαθμολογίες για κάθε χρήστη σε ένα λεξικό.

Για το σύστημα προτάσεων χρησιμοποιουμε συνεργατικό φιλτράρισμα(collaborative filtering), όπου το σύστημα προτείνει στοιχεία σε έναν χρήστη με βάση τις προτιμήσεις και τις συμπεριφορές άλλων χρηστών που έχουν παρόμοιες προτιμήσεις. Συγκεκριμένα, το σύστημα χρησιμοποιεί μια παραλλαγή συνεργατικού φιλτραρίσματος που ονομάζεται k-πλησιέστεροι γείτονες (k-NN) με ένα νευρωσικό δίκτυο για να προβλέψει τις αξιολογήσεις των χρηστών.

Σε αυτό το κομμάτι του κώδικα εκτελούμε ομαδοποίηση χρηστών με βάση τις αξιολογήσεις ταινιών τους χρησιμοποιώντας τον πίνακα ομοιότητας συνημίτονου και τον αλγόριθμο MiniBatchKMeans από τη βιβλιοθήκη scikit-learn.

Στον κώδικα δημιουργούμε πρώτα μια μήτρα στοιχείου χρήστη με βάση το DataFrame 'filtered_df', η οποία περιέχει πληροφορίες σχετικά με τους χρήστες, τις ταινίες και τις αντίστοιχες αξιολογήσεις τους. Αυτός ο πίνακας είναι ένας αραιός πίνακας, ο οποίος αναπαρίσταται σε μορφή COO χρησιμοποιώντας τη συνάρτηση 'coo_matrix()'.

Στη συνέχεια, ο πίνακας ομοιότητας συνημίτονου υπολογίζεται από τον πίνακα στοιχείων χρήστη χρησιμοποιώντας τη συνάρτηση 'cosine_similarity()' από τη βιβλιοθήκη scikit-learn. Αυτός ο πίνακας περιέχει τις βαθμολογίες ομοιότητας κατά ζεύγη συνημίτονου μεταξύ κάθε ζεύγους χρηστών στο σύνολο δεδομένων.

Στη συνέχεια, χρησιμοποιούμε τον αλγόριθμος MiniBatchKMeans για την ομαδοποίηση των χρηστών σε «n_clusters» αριθμό συμπλεγμάτων. Ο αλγόριθμος αντιστοιχίζει σε κάθε χρήστη ένα σύμπλεγμα με βάση τις βαθμολογίες ομοιότητας συνημίτονου τους, οι οποίες αντιπροσωπεύουν την ομοιότητά τους όσον αφορά τις προτιμήσεις ταινιών τους. Η μεταβλητή 'user_cluster_labels' αποθηκεύει την ετικέτα συμπλέγματος που έχει εκχωρηθεί σε κάθε χρήστη.

Στη συνέχεια, ομαδοποιούμε τους χρήστες σε συμπλέγματα με βάση τις ετικέτες συμπλέγματος που τους έχουν εκχωρηθεί. Η μεταβλητή 'user_clusters' είναι μια λίστα συνόλων, όπου κάθε σύνολο περιέχει τους δείκτες των χρηστών στο αντίστοιχο σύμπλεγμα.

Τέλος, εκτυπώνουμε τον αριθμό των χρηστών σε κάθε σύμπλεγμα χρησιμοποιώντας έναν βρόχο for που επαναλαμβάνεται στον αριθμό των συμπλεγμάτων.

Δοκιμή του Κώδικα

Όταν τρέχουμε το main.py παίρνουμε αυτό:

```
Select an option:
1. Print number of entries, unique users, and unique movies
2. Print filtered DataFrame for valid users
3. Print user count by vote count
4. Plot histogram of number of votes per user
5. Plot histogram of temporal range of reviews per user
6. Create an alternative representation of the dataset as preference
vectors
7. Cluster the limited set of users into L clusters using euclidian
8. Cluster the limited set of users into L clusters cosine
9. Visualize the clusters
10. Cluster the limited set of users into L clusters
11.create a collaborative filtering recommendation system using a
neural network
0. Exit
```

Πατώντας 1 παίρνουμε:

```
Number of entries:  4669820
Number of unique users:  1499238
Number of unique movies:  351109
```

Πατώντας 2 παίρνουμε:

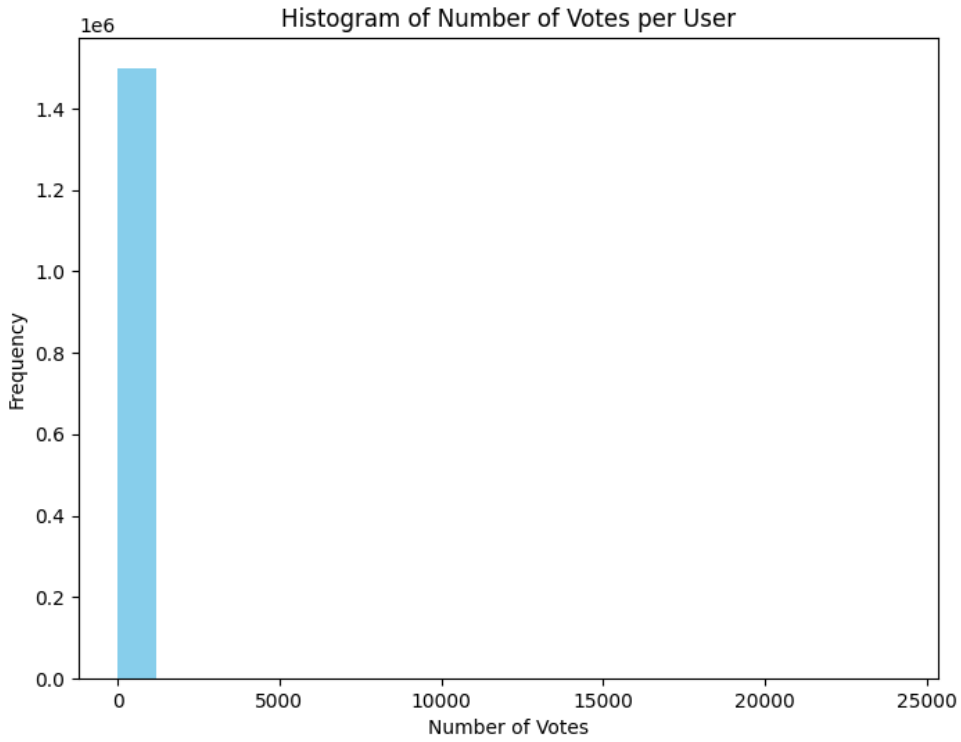
```
            userID    movieID rating      review_date
2        ur3780035  tt0387887      8  16 January 2005
9        ur2694867  tt0060196     10  16 January 2005
12       ur4318504  tt0113724      7  16 January 2005
14       ur4545306  tt0325596      8  16 January 2005
29       ur2694867  tt0071385      1  16 January 2005
...            ...        ...    ...              ...
4669789  ur4592163  tt0104031     10  16 January 2005
4669793  ur1826984  tt0086006      2  16 January 2005
4669802  ur3885602  tt0312549      8  16 January 2005
4669805  ur1596522  tt0274711      1  16 January 2005
4669818  ur4581944  tt0102614      8  16 January 2005

[901720 rows x 4 columns]
```

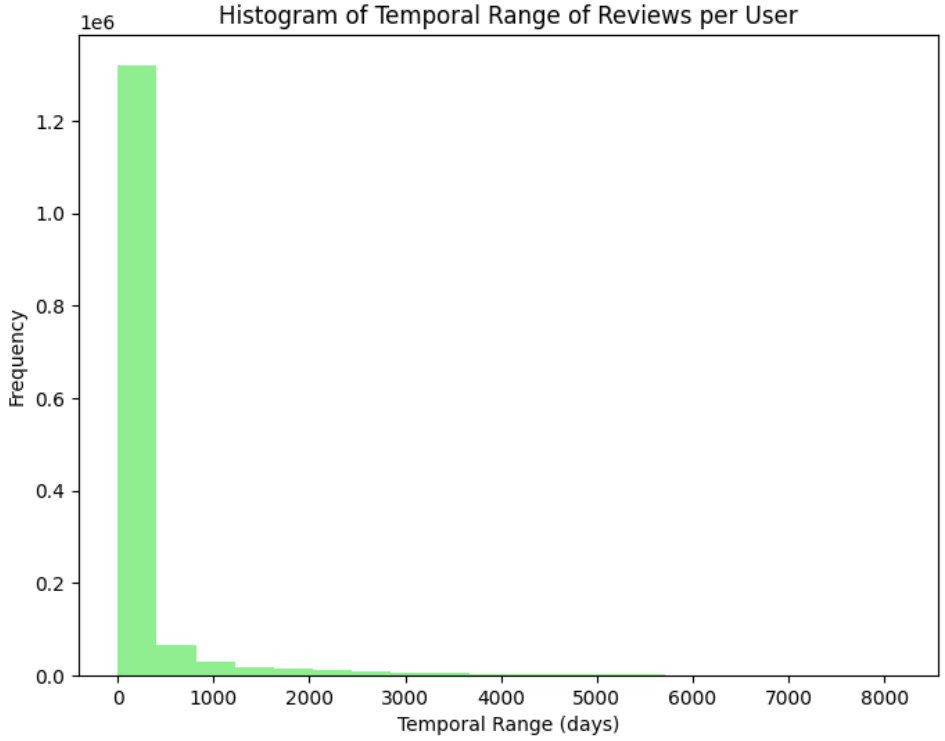
Πατώντας 3 παίρνουμε:

```
1       1069533
2        191934
3         76087
4         41434
5         25671
         ...
691           1
1772          1
677           1
1617          1
1006          1
Name: count, Length: 905, dtype: int64
```

Πατώντας 4 παίρνουμε:



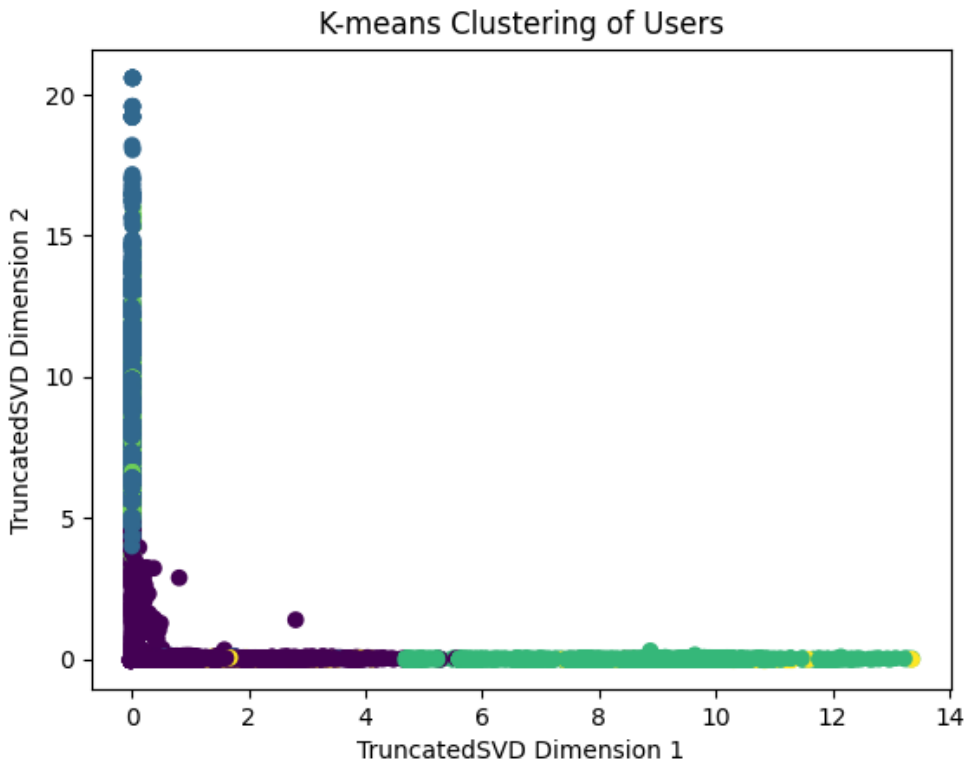
Πατώντας 5 παίρνουμε:



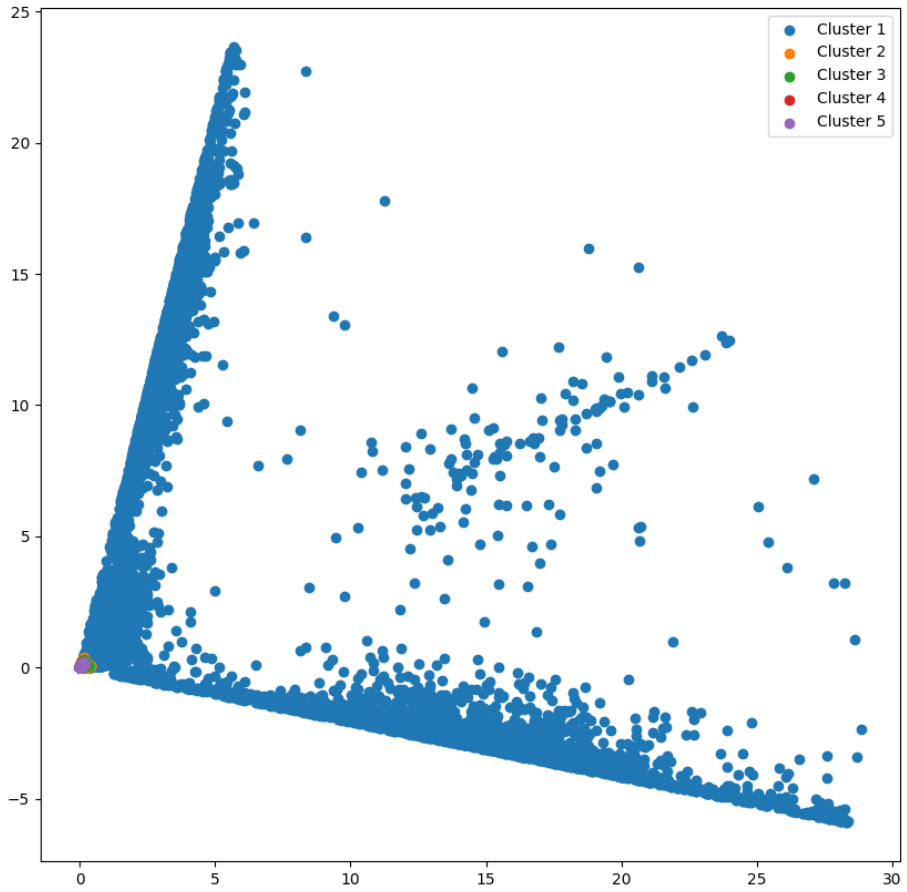
Πατώντας 6 παίρνουμε:

```
(1285609, 287730)      9
  (1285609, 313250)    6
  (1285609, 320136)    10
  (1285616, 158423)    9
  (1285616, 170236)    3
  (1285616, 258496)    9
  (1285616, 320854)    1
  (1285616, 335212)    10
  (1285616, 340544)    10
  (1285617, 231404)    8
  (1285617, 314529)    10
  (1285617, 315202)    1
  (1285617, 322514)    4
  (1285617, 330255)    10
...
  (1498681, 6976)      7
  (1498701, 675)       8
  (1498701, 4885)      7
  (1498701, 5496)      7
  (1498701, 6884)      9
  (1498747, 3)         4
  (1498747, 50)        9
  (1498747, 95)        5
  (1498868, 2527)      9
  (1498868, 14275)     10
  (1498868, 67634)     7
  (1498970, 2462)      10
  (1498970, 3173)      8
  (1498970, 10272)     9
  (1498970, 56858)     9
```

Πατώντας 7 και 9 παίρνουμε:



Πατώντας 8 και 9 παίρνουμε:



Αφού η απεικόνιση των συστάδων εμφανίζεται ως σχήμα "L" με τα περισσότερα σημεία να συγκεντρώνονται γύρω από τους άξονες x και y, παρόλο που υπάρχουν μερικές εξαιρέσεις, υποδηλώνει ότι τα συμπλέγματα είναι σχετικά καλά διαχωρισμένα και έχουν διακριτά όρια.

Τα συμπλέγματα που συγκεντρώνονται κοντά στους άξονες x και y υποδεικνύουν ότι αυτές οι ομάδες σημείων δεδομένων έχουν παρόμοια χαρακτηριστικά ή προτιμήσεις, καθώς βρίσκονται κοντά μεταξύ τους στον μειωμένο διαστατικό χώρο.

Οι εξαιρέσεις ή τα σημεία δεδομένων που δεν συγκεντρώνονται στενά με τα περισσότερα σημεία, ενδέχεται να αντιπροσωπεύουν ακραίες τιμές ή περιπτώσεις που έχουν μοναδικά χαρακτηριστικά που δεν ευθυγραμμίζονται με την πλειονότητα των σημείων δεδομένων σε ένα συγκεκριμένο σύμπλεγμα. Αυτές οι ακραίες τιμές μπορεί να είναι ενδιαφέρουσες περιπτώσεις που δικαιολογούν περαιτέρω διερεύνηση ή μπορεί να υποδεικνύουν την ανάγκη βελτίωσης της προσέγγισης ομαδοποίησης.

Συνοπτικά, το σχήμα υποδηλώνει ότι ο αλγόριθμος ομαδοποίησης έχει αναγνωρίσει με επιτυχία διακριτές ομάδες με βάση ομοιότητες ή προτιμήσεις, αλλά υπάρχουν ορισμένα

σημεία δεδομένων που αποκλίνουν από τα περισσότερα, πιθανώς λόγω μοναδικά χαρακτηριστικά ή χαρακτηριστικά.

Επειδή για τα 10 και 11 έπαιρναν παραπάνω ώρα με το dataset θα χρησιμοποιήσουμε το newdataset

Πατώντας 10 παίρνουμε:

```
Enter your choice (1-11): 10
[[0.   1.   1.   ... 1.   1.   1.  ]
 [1.   0.   1.   ... 1.   1.   1.  ]
 [1.   1.   0.   ... 1.   1.   1.  ]
 ...
 [1.   1.   1.   ... 0.   0.86 0.92]
 [1.   1.   1.   ... 0.86 0.   0.94]
 [1.   1.   1.   ... 0.92 0.94 0.  ]]
```

Πατώντας 11 παίρνουμε:

```
Epoch 1/10
192/192 [==============================] - 1s 1ms/step - loss:
0.0000e+00
Epoch 2/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 3/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 4/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 5/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 6/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 7/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 8/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 9/10
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
Epoch 10/10
```

```
192/192 [==============================] - 0s 1ms/step - loss:
0.0000e+00
192/192 [==============================] - 0s 847us/step
Epoch 1/10
1/1 [==============================] - 1s 601ms/step - loss:
0.0000e+00
Epoch 2/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 3/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 4/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 5/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 6/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 7/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 8/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 9/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
Epoch 10/10
1/1 [==============================] - 0s 2ms/step - loss: 0.0000e+00
1/1 [==============================] - 0s 47ms/step
Epoch 1/10
16/16 [==============================] - 1s 2ms/step - loss: 0.0149
Epoch 2/10
16/16 [==============================] - 0s 2ms/step - loss: 0.0078
Epoch 3/10
16/16 [==============================] - 0s 1ms/step - loss: 0.0032
Epoch 4/10
16/16 [==============================] - 0s 2ms/step - loss: 0.0067
Epoch 5/10
16/16 [==============================] - 0s 1ms/step - loss: 0.0031
Epoch 6/10
16/16 [==============================] - 0s 1ms/step - loss: 3.6277e-
04
Epoch 7/10
16/16 [==============================] - 0s 1ms/step - loss: 0.0035
Epoch 8/10
16/16 [==============================] - 0s 1ms/step - loss: 0.0058
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.0025
Epoch 10/10
16/16 [==============================] - 0s 2ms/step - loss: 0.0029
16/16 [==============================] - 0s 2ms/step
```

Άρα το νευρωνικό δίκτυο φαίνεται να δουλεύει.