ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Αναγνώριση Προτύπων

Εργασία 2018-2019

Π16036 – Ιωαννίδης Παναγιώτης
Π16112 – Παραβάντης Αθανάσιος

Περιεχόμενα

1 Εισαγωγή

Η παρούσα εργασία για το μάθημα της Αναγνώρισης Προτύπων βασίζεται στο σύνολο δεδομένων MovieLens 100K Dataset που είναι διαθέσιμο στην ιστοσελίδα του Grouplens. Για τη επίλυση όλων των ερωτημάτων της εργασίας, χρησιμοποιήσαμε τη Python με την έκδοση 3.7.2, καθώς και αρκετές βιβλιοθήκες ανάλυσης δεδομένων και μηχανικής μάθησης.

Η λίστα με τις βιβλιοθήκες της Python που χρησιμοποιούνται στην εργασία είναι οι εξής:

- Pandas
  Βιβλιοθήκη που παρέχει δομές και εργαλεία για την ανάλυση δεδομένων.
- Scikit Learn
  Βιβλιοθήκη μηχανικής μάθησης αλλά και ανάλυσης δεδομένων.
- Scipy
  Σύνολο βιβλιοθηκών που παρέχει βασικές δομές για τη κάλυψη μαθηματικών υπολογισμών.
- Numpy
  Βιβλιοθήκη για επιστημονικούς υπολογισμούς, κυρίως πινάκων.
- Matplotlib
  Βιβλιοθήκη για την προβολή δεδομένων με γραφικές παραστάσεις.

Δόθηκε απάντηση για όλα τα ερωτήματα της εργασίας.

2 Ερώτημα 1

Αφού διαβάσαμε το αρχείο README και κατανοήσαμε τη δομή των δεδομένων αποφασίσαμε να δημιουργήσουμε τη κλάση **MovieLensData** στο αρχείο **movielens_data.py** ώστε να διαχειριζόμαστε τα αρχεία αυτά και να λαμβάνουμε, κάθε φορά, τα δεδομένα που μας είναι απαραίτητα.

Συγκεκριμένα, γίνεται η εξής επεξεργασία και φόρτωση δεδομένων:

- Σαρώνονται όλα τα περιεχόμενα του αρχείου u.data με τη βοήθεια της συνάρτηση read_csv του pandas και αποθηκεύονται στη μνήμη οι βαθμολογίες χρηστών που είναι μεγαλύτερες από 4.
- Σαρώνονται τα περιεχόμενα του αρχείου u.item ώστε να τοποθετήσουμε στη μνήμη πληροφορίες για κάθε ταινία.
- Ενοποίηση των βαθμολογιών μαζί με τις πληροφορίες για κάθε ταινία, με βάση το user id των δεδομένων.
- Εύρεση του αθροίσματος ταινιών ανά κατηγορία στις οποίες ο χρήστης έχει βάλει βαθμό.
- Κανονικοποίηση των δεδομένων με βάση τον μέσο όρο του αθροίσματος ταινιών ανά κατηγορία για κάθε στήλη.

Η παραπάνω προεπεξεργασία και φόρτωση δεδομένων είναι πολύτιμη επειδή χρησιμοποιείται ως είσοδο για τα επόμενα ερωτήματα της εργασίας. Οποιαδήποτε σάρωση και αναπαράσταση δεδομένων γίνεται με τη βοήθεια των δομών της βιβλιοθήκης pandas.

3 Ερώτημα 2

3.1 Ζητούμενο

Εφαρμόστε το βασικό σχήμα ακολουθιακής ομαδοποίησης για να εκτιμήσετε το πλήθος των ομάδων των χρηστών, ως προς τις προτιμήσεις τους.

3.2 Τεκμηρίωση

3.2.1 Βασικό ακολουθιακό αλγοριθμικό σχήμα (BSAS)

Στο ερώτημα αυτό, υλοποιήσαμε το βασικό ακολουθιακό αλγοριθμικό σχήμα (BSAS) όπως και τον αλγόριθμο εκτίμησης αριθμού ομάδων.

Στην περίπτωση μας ο αριθμός των ομάδων δεν είναι γνωστός. Το ζητούμενο μας είναι να βρούμε τον αριθμό αυτόν. Εφόσον δεν γνωρίζουμε τον αριθμό των ομάδων, η παράμετρος η οποία εμπλέκεται στον αλγόριθμο και η οποία ορίζεται από τον χρήστη, είναι το κατώφλι ανομοιότητας Θ.

Κάθε νέο διάνυσμα εξετάζεται από τον αλγόριθμο και καταχωρείται είτε σε μια από τις υπάρχουσες ομάδες είτε σε μια νέα ομάδα που δημιουργείται, ανάλογα με την απόσταση από τις άλλες υπάρχουσες ομάδες.

Για τον υπολογισμό της απόστασης θα χρησιμοποιήσουμε το τετράγωνο της ευκλείδειας απόστασης.

Η διάταξη με την οποία παρουσιάζονται τα διανύσματα στον BSAS παίζει σημαντικό ρόλο στη διαδικασία ομαδοποίησης, καθώς μπορεί να οδηγήσουν σε διαφορετικές ομαδοποιήσεις. Επίσης σημαντικός παράγοντας είναι και το Θ.

Κάθε ομάδα αντιπροσωπεύεται από ένα διάνυσμα. Ο αντιπρόσωπος τις ομάδας μετρά από κάθε προσθήκη ενός νέου διανύσματος στην ομάδα θα πρέπει να ενημερώνεται.

3.2.2 Εκτίμηση αριθμού ομάδων

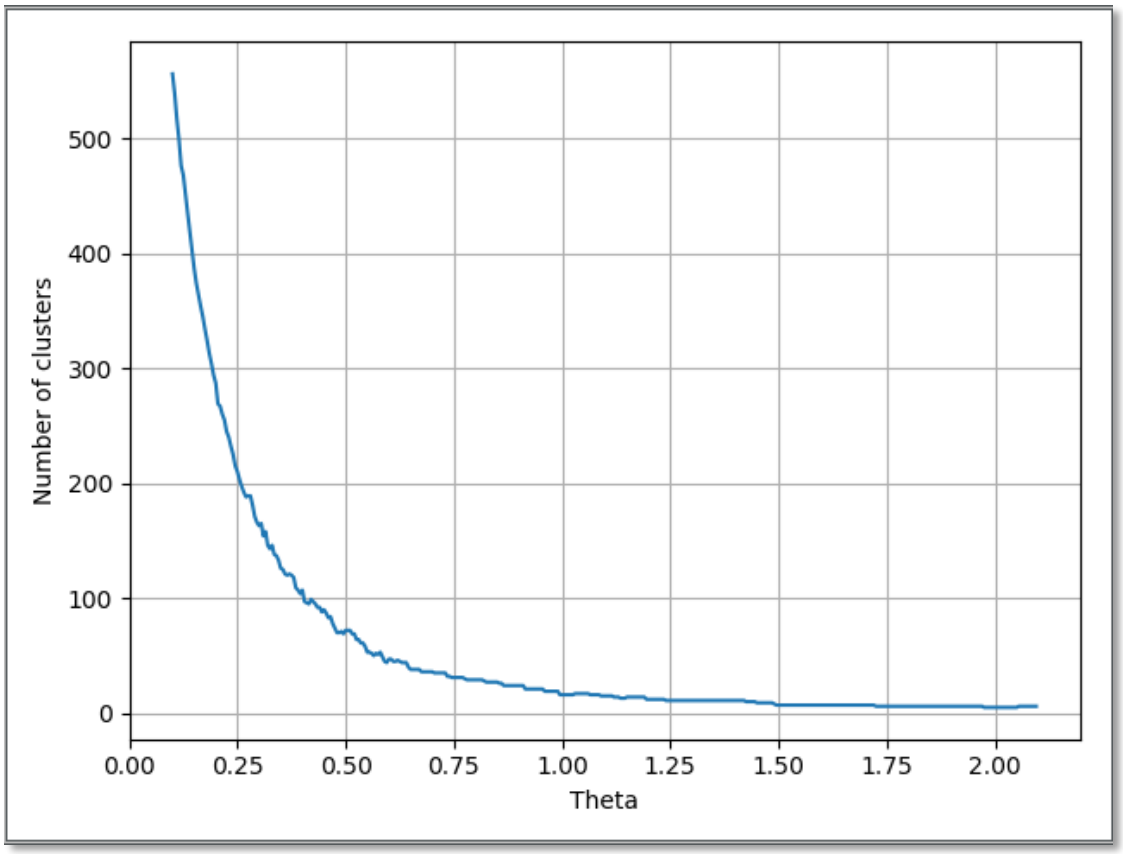
Για να μπορέσουμε να εκτιμήσουμε τον αριθμό των ομάδων, αρκεί να εκτελέσουμε επαναληπτικά τον αλγόριθμο BSAS, για Θ από a έως b με βήμα c. Όσο περισσότερες φόρες εκτελέσουμε τον αλγόριθμο BSAS τόσο μεγαλύτερη θα είναι και η ακρίβεια των αποτελεσμάτων.

3.2.3 Εκτέλεση

Ο αλγόριθμος BSAS και αλγόριθμος εκτίμησης αριθμού ομάδων βρίσκονται στο αρχείο **bsas.py**. Ως ελάχιστο Θ δίνουμε την τιμή 0.1. ως μέγιστο Θ την τιμή 2.1 και ως βήμα την τιμή 0.005. Τα δεδομένα που δίνονται ως είσοδος στο αλγόριθμο είναι το άθροισμα των κατηγοριών των ταινιών με βαθμολογία από 4 και πάνω. Στη συνέχεια κανονικοποιούμε τα διανύσματα που προκύπτουν ώστε οι τιμές να είναι μεταξύ 0 και 1.

Αρκεί να εκτελέσουμε το αρχείο **bsas.py** και στο αρχείο result.txt που θα δημιουργηθεί θα δούμε τον αριθμό των ομάδων που δημιουργούνται για κάθε Θ.

Μετά την εκτέλεση του αλγόριθμου προκύπτει η ακόλουθη γραφική απεικόνιση του αριθμού των ομάδων ως προς Θ, εικόνα 3.1.



Εικόνα 3.1

3.2.4 Συμπέρασμα

Με βάση τις παραμέτρους που έχουμε θέσει ο αριθμός των ομάδων είναι n = 7.

4 Ερώτημα 3

4.1 Ζητούμενο

Στο 3ο ερώτημα ζητείται η εφαρμογή του αλγόριθμου K-Μέσων και του αλγορίθμου ιεραρχικής ομαδοποίησης, πάνω στο dataset της εργασίας.

4.2 Αλγόριθμος K-Μέσων

Ο αλγόριθμος K-Μέσων υλοποιείται από τη βιβλιοθήκη SciKit Learn, χρησιμοποιώντας τη κλάση KMeans. Η υλοποίηση του ερωτήματος είναι στο αρχείο **kmeans.py**. Το σύνολο δεδομένων, η προεπεξεργασία και η κανονικοποίησή τους γίνεται με τη βοήθεια της κλάσης MovieLensData.

Επιλογές παραμέτρων για τον αλγόριθμο:

- Ο αριθμός των ομάδων και των κεντροειδών επιλέξαμε να είναι 7.
- Ο αριθμός των επαναλήψεων επιλέξαμε να είναι 100.

Τα δεδομένα που εισάγονται στον αλγόριθμο είναι 18 διαστάσεων και αναπαριστούν το άθροισμα των βαθμολογιών ανά κατηγορία για κάθε χρήστη. Οι διαστάσεις αυτές προκύπτουν από τις 18 κατηγορίες ταινιών στο σύνολο δεδομένων που μας παρέχεται. Εκτελώντας το αρχείο **kmeans.py** λαμβάνουμε ενδεικτικά αποτελέσματα για τα κεντροειδή καθώς και την ομαδοποίηση που προκύπτει.

Παρακάτω εμφανίζονται οι συντεταγμένες των 7 κεντροειδών με (x_0, x_1, …, x_{17}), εικόνα 4.1, να τονίσουμε ότι ο πίνακας παρουσιάζεται σε τρία κομμάτια ώστε τα δεδομένα του ευανάγνωστα:

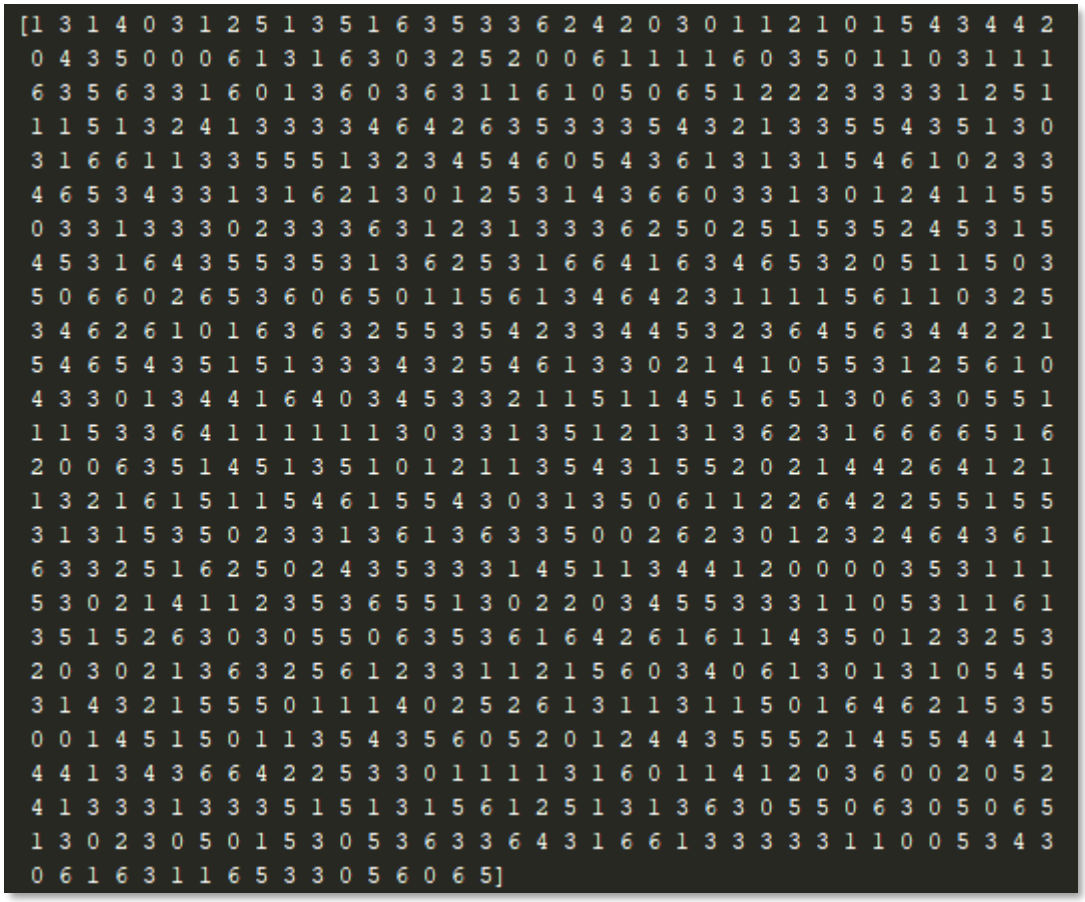
0	1	2	3	4	5
0.802310708815007	0.5701816924479384	0.18890341465055638	0.3125954978467549	0.8473790581458686	0.11237606593665406
0.16253777480504716	0.08890151425102899	0.024255915803604394	0.04950625870049252	0.34267153185879484	0.14691923550859298
0.3285361257825508	0.18883311119325719	0.09097709486202239	0.14810083102821375	0.863812821977405	0.13006845659384825
0.5619751991527961	0.1543352779028949	0.030502907224530515	0.07943351031176013	0.27939100315121257	0.2520175100303283
0.7961964129073305	0.3700421835808827	0.08255507997185974	0.13812189431840569	0.6908199297933243	0.24862382196041866
0.44178438786283214	0.19416027709049868	0.04786053890697679	0.07835056952187366	0.40693013870752115	0.2299188940086565
0.9850302408914853	0.5323783621104178	0.05816284137889969	0.08434083058517876	0.32622659460822584	0.15924788203021883

6	7	8	9	10	11
0.008884556828680877	0.8186939990843385	0.03660226767944748	0.02026203713762377	0.0789891427626968	0.19642257874578678
0.017005372388424336	0.998263888888889	0.006870538866846862	0.0386404130760519	0.022451221330091586	0.04579656145662579
0.01807610765873934	0.8397548307356346	0.019441686211575333	0.029597546563432997	0.05917471639327429	0.11397514470404095
0.013182722535924818	0.7357944799569105	0.008026163573798768	0.07258339583465859	0.29611744557440217	0.05206579782039517
0.00899222704272988	0.9524553831879718	0.0188994373270453	0.04173684985964771	0.13614377337907133	0.08571109076986344
0.018018076060788483	0.9987922705314011	0.009624688584372338	0.06730522775710224	0.07808214567683024	0.06712050547488334
0.004484872012045933	0.4878505893240153	0.019881864293431754	0.024639076288308477	0.12115480557339489	0.05051362638077693

12	13	14	15	16	17
0.0900440413450603	0.6044105681123865	0.48537406641354114	0.4285663335196577	0.31870158008389426	0.041548931078704304
0.09920816480266209	0.3564454286915804	0.09223269274451287	0.2268691716931936	0.17253411786693323	0.019468650206772006
0.09553837473599006	0.5649289406070116	0.18564288562435127	0.30028080117483485	0.16834926014139653	0.02780313274276394
0.32711404976257474	0.31557595257418614	0.22959436862926305	0.946325200439369	0.1566313070889635	0.006961623242536359
0.144086557001022	0.49817865478489637	0.39914754987963463	0.702587264680129	0.3065083242871761	0.0358430926888618
0.16258909906162816	0.3721189441360514	0.23162535693751576	0.4918232185363005	0.25414531506651467	0.02784001032272067
0.08375145722704123	0.3692475659251296	0.5468953824214156	0.556962345797337	0.29917985658744484	0.02358806176128727

Εικόνα 4.1 - Οι πρώτες γραμμές αντιστοιχούν στην πρώτη ομάδα, οι δεύτερες στην δεύτερη ομάδα και ούτω καθεξής.

Επιπλέον, παρατηρούμε την ομαδοποίηση που γίνεται για στοιχεία του πίνακα, εικόνα 4.2.



Εικόνα 4.2

Ο αριθμός που προκύπτει από το 0 έως το 6 βασίζεται στην αρχική επιλογή των 7 ομάδων που ορίσαμε ως παράμετρο στη κλάση της βιβλιοθήκης. Τα αποτελέσματα εμφανίζονται έπειτα από 100 επαναλήψεις του αλγορίθμου.

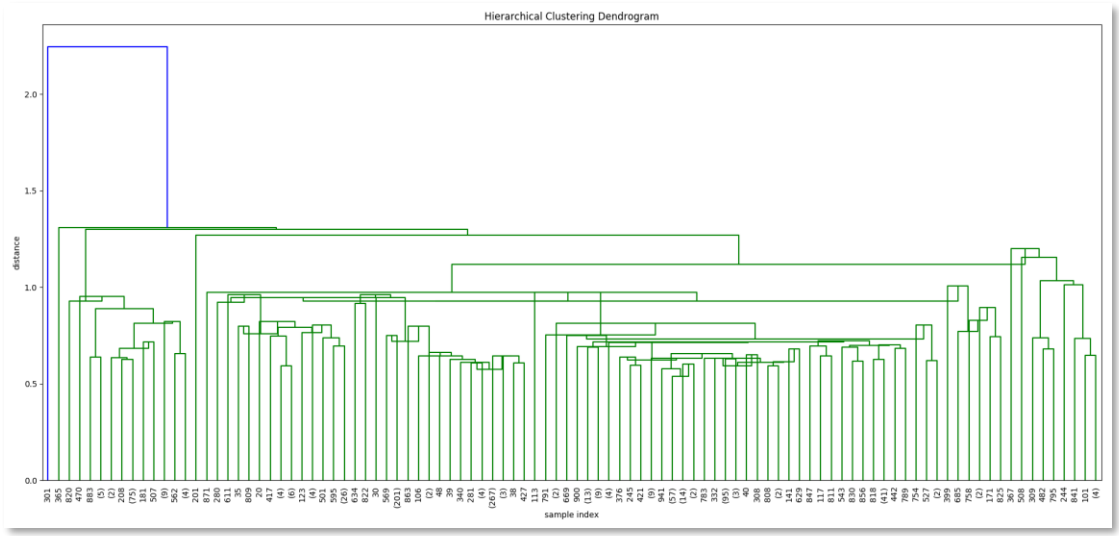
4.3 Αλγόριθμος Ιεραρχικής Ομαδοποίησης

Ο αλγόριθμος ιεραρχικής ομαδοποίησης υλοποιείται από τη βιβλιοθήκη Scipy, χρησιμοποιώντας τη συνάρτηση linkage. Η υλοποίηση του ερωτήματος είναι στο αρχείο **hierarchical.py**. Το σύνολο δεδομένων, η προεπεξεργασία και η κανονικοποίησή τους γίνεται με τη βοήθεια της κλάσης MovieLensData.

Όπως και προηγουμένως, τα δεδομένα μας είναι σε 18 διαστάσεις λόγω του πλήθους των κατηγοριών για τις ταινίες. Οι παράμετροι του αλγορίθμου που καθορίσαμε είναι οι εξής:

- Επιλέγουμε τη μέθοδο ‘median’ για την ενοποίηση των κεντροειδών. Αυτό σημαίνει ότι ο αλγόριθμος υπολογίζει τη μέση τιμή μεταξύ των κεντροειδών δυο ομάδων.
- Επιλέγουμε την ευκλείδεια απόσταση για τον καθορισμό του δείκτη ομοιότητας μεταξύ των δεδομένων μας.

Με την εκτέλεση του αρχείου **hierarchical.py** παίρνουμε το ακόλουθο δενδρόγραμμα που αναπαριστά τη διαδικασία του αλγορίθμου ενοποιώντας τις ομάδες:



Εικόνα 4.3

4.4 Συμπέρασμα

Έχοντας υπ' όψη τους δυο αλγόριθμους ομαδοποίησης Κ-Μέσων και Ιεραρχικής Ομαδοποίησης παρατηρούμε ότι ενώ χρησιμοποιούνται για τον ίδιο σκοπό, το αποτέλεσμα είναι αρκετά διαφορετικής σημασίας.

Στον αλγόριθμο Κ-Μέσων οι επαναλήψεις οδηγούν στην ομαδοποίηση όλων των δεδομένων σε μια από τις κατηγορίες που ορίζουμε. Όταν όμως μελετάμε την ιεραρχική ομαδοποίηση, παρατηρούμε ότι το δενδρόγραμμα παρέχει πληροφορίες ιεραρχίας μεταξύ των δεδομένων και όχι μια απλή ομαδοποίηση. Αυτό το στοιχείο μπορεί να χρησιμοποιηθεί για ταξινόμηση δεδομένων βιολογικής σημασίας, όμως στη περίπτωσή μας το σύνολο δεδομένων είναι αρκετά διαφορετικό.

5 Ερώτημα 4

5.1 Ζητούμενο

Στο 4ο ερώτημα μας ζητείται η υλοποίηση δυο ταξινομητών, ο ένας πάνω στο μοντέλο των νευρωνικών δικτύων (multi perceptron) και ο άλλος πάνω στο μοντέλο των ελαχίστων τετραγώνων (least squares). Δοθείσης ενός χρήστη και μίας ταινίας, καλούμαστε να εκπαιδεύσουμε κατάλληλα τους ταξινομητές ώστε να προβλέπουν εάν ο χρήστης έχει δει τη ταινία.

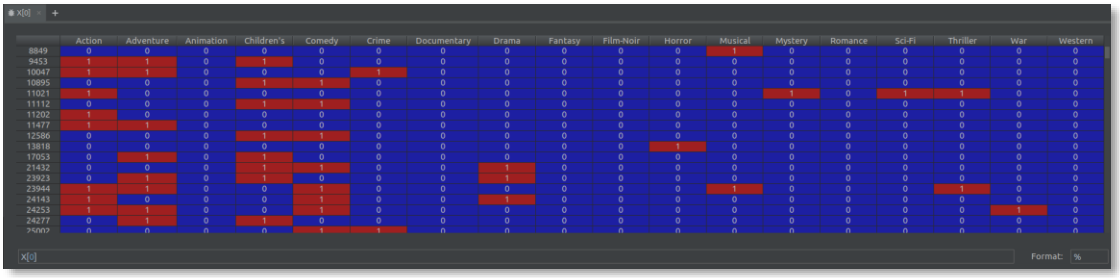
5.2 Αλγόριθμος Ελαχίστων Τετραγώνων

Για να υλοποιήσουμε τον αλγόριθμο least squares, χρησιμοποιούμε τη κλάση LinearRegression της βιβλιοθήκης SciKit Learn. Η υλοποίηση του ερωτήματος είναι στο αρχείο **least_squares.py**.

Η μορφή των δεδομένων της εκπαίδευσης είναι το σύνολο των ταινιών που έχει δει ο κάθε χρήστης και περιλαμβάνει τις κατηγορίες στις οποίες ανήκει κάθε ταινία. Για να επιλέξουμε ποια αρχεία από το 5-fold σχήμα θέλουμε να χρησιμοποιήσουμε αρκεί να αλλάξουμε την τιμή της μεταβλητής **fold_number.** Εμείς, για τα αποτελέσματα που παρουσιάζονται στην εργασία έχουμε επιλέξει το 5.

Το σύνολο των ετικετών περιλαμβάνει τιμές 1 εάν έχει βαθμολογήσει ο χρήστης τη ταινία με βαθμό μεγαλύτερο του 3 και -1 όταν η βαθμολογία είναι μικρότερη ή ίση αυτού.

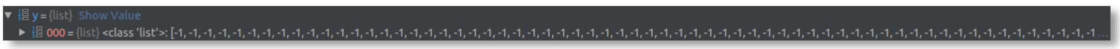
Ενδεικτικά, το πρώτο στοιχείο του Χ είναι ένα σύνολο δεδομένων 18 διαστάσεων για τον πρώτο χρήστη και το μέγεθός του είναι ο αριθμός των ταινιών που έχει βαθμολογήσει ο χρήστης, εικόνα 5.1:



	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
8849	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
9453	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10047	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
10895	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11021	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
11112	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11202	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11477	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12586	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13818	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
17053	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21432	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
23923	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
23944	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
24143	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
24253	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
24277	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25002	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Εικόνα 5.1

Οι ετικέτες για τον πρώτο χρήστη για τις ταινίες που έχει βαθμολογήσει είναι οι εξής:



Εικόνα 5.2

Ο λόγος για τον οποίο επιλέγουμε αυτή τη δομή των δεδομένων προς εκπαίδευση βασίζεται στο γεγονός ότι προσπαθούμε να συσχετίσουμε κάθε ταινία και βαθμολογία με τις κατηγορίες στις οποίες ανήκει. Έτσι, όταν καλείται ο αλγόριθμος least squares να πάρει απόφαση για μια άλλη ταινία του ίδιου χρήστη, το αποτέλεσμα βασίζεται στις κατηγορίες των ταινιών και τους βαθμούς που έχει δώσει στο παρελθόν.

Για την αποτελεσματική εφαρμογή του αλγόριθμο, δημιουργούμε μια νέα κλάση LinearRegression για κάθε χρήστη, έτσι ώστε τα δεδομένα εκπαίδευσης και οι ετικέτες να αφορούν τον ίδιο τον χρήστη και μόνο.

Εκτελώντας το αρχείο **least_squares.py**, κάνουμε ερώτηση για τον χρήστη 1 εάν έχει δε τη ταινία 4. Σε αυτή τη περίπτωση λαμβάνουμε θετική απάντηση από τον ταξινομητή:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
In[2]: runfile('C:\\Users\\pioan\\PycharmProjects\\pattern_recognition\\least_square.py', wdir='C:
Give a user id (1-943):>? 1
Give a movie id (1-1682)>? 4
The user has seen this movie
Give a user id (1-943):
>?
```

Εικόνα 5.3

Εάν κάνουμε μια άλλη ερώτηση για τον χρήστη 1 και τη ταινία 520, λαμβάνουμε αρνητική απάντηση από τον ταξινομητή:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
In[2]: runfile('C:\\Users\\pioan\\PycharmProjects\\pattern_recognition\\least_square.py', wdir='C:
Give a user id (1-943):>? 1
Give a movie id (1-1682)>? 520
The user has NOT seen this movie
Give a user id (1-943):
>?
```

Εικόνα 5.4

Ερώτηση id χρήστη και id ταινίας γίνεται επαναληπτικά ώστε να μπορούν να γίνουν πολλές δοκιμές χωρίς να είναι απαραίτητη η επανεκτέλεση του αρχείου.

5.3 Αλγόριθμος Perceptron

Για να υλοποιήσουμε τον αλγόριθμο multilayer perceptron, χρησιμοποιούμε τη κλάση MLPClassifier της βιβλιοθήκης SciKit Learn. Η υλοποίηση του ερωτήματος είναι στο αρχείο **multilayer_perceptron.py**.

Όπως και στην υλοποίηση του ταξινομητή least squares, έτσι και σε αυτή τη περίπτωση, τα δεδομένα μας αφορούν όλες τις ταινίες που έχει βαθμολογήσει κάθε χρήστης μαζί με τις κατηγορίες στις οποίες ανήκει η κάθε ταινία. Για να επιλέξουμε ποια αρχεία από το 5-fold σχήμα θέλουμε να χρησιμοποιήσουμε αρκεί να αλλάξουμε την τιμή της μεταβλητής **fold_number.** Εμείς, για τα αποτελέσματα που παρουσιάζονται στην εργασία έχουμε επιλέξει το 5.

Οι ετικέτες λαμβάνουν τιμές 1 και -1, εάν οι χρήστης έχει θέσει βαθμολογία μεγαλύτερη ή μικρότερη-ίση του 3 αντίστοιχα.

Τα χαρακτηριστικά του νευρωνικού δικτύου επιλέξαμε να είναι τα εξής:

- 3 hidden layers, 9 neurons για το επίπεδο 1, 5 neurons για το επίπεδο 2 και 9 neurons για το επίπεδο 3
- Συνάρτηση ενεργοποίησης logistic sigmoid function.
- Μέθοδος weight optimization lbfgs optimizer.

Ακολουθώντας το ίδιο σκεπτικό στην υλοποίηση του προγράμματος, εκτελούμε το αρχείο **multilayer_perceptron.py**. Εάν κάνουμε ερώτηση για τον χρήστη 1 και την ταινία 4 παίρνουμε θετική απάντηση από τον ταξινομητή:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
In[2]: runfile('C:/Users/pioan/PycharmProjects/pattern_recognition/multilayer_perceptron.py', wdir
Give a user id (1-943):>? 1
Give a movie id (1-1682)>? 4
The user has seen this movie
Give a user id (1-943):
>?
```

Εικόνα 5.5

Εάν κάνουμε ερώτηση για τον χρήστη 1 και την ταινία 520 παίρνουμε αρνητική απάντηση από τον ταξινομητή:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
In[2]: runfile('C:/Users/pioan/PycharmProjects/pattern_recognition/multilayer_perceptron.py', wdir
Give a user id (1-943):>? 1
Give a movie id (1-1682)>? 520
The user has NOT seen this movie
Give a user id (1-943):
>?
```

Εικόνα 5.6

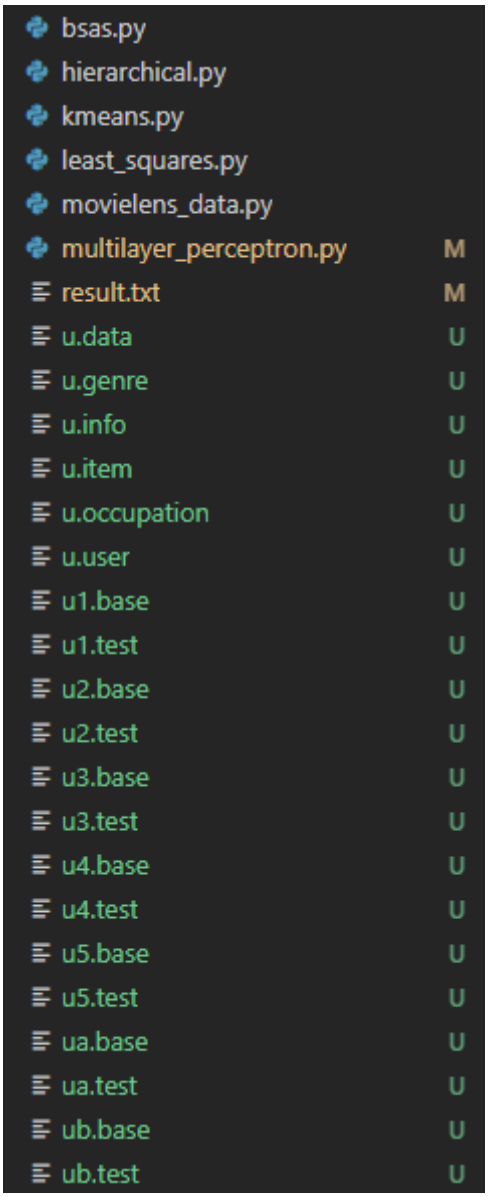
Ερώτηση id χρήστη και id ταινίας γίνεται επαναληπτικά ώστε να μπορούν να γίνουν πολλές δοκιμές χωρίς να είναι απαραίτητη η επανεκτέλεση του αρχείου.

5.4 Συμπέρασμα

Παρατηρώντας τα αποτελέσματα των δυο ταξινομητών, παρατηρούμε ότι βασιζόμενοι στο σύνολο δεδομένων 5-fold που μας παρέχεται από το dataset, μπορούμε να εξάγουμε συμπεράσματα και προβλέψεις για τις ταινίες που προτιμούν οι χρήστες. Τόσο ο αλγόριθμος least squares όσο και ο perceptron, ενώ έχουν σημαντικές διαφορές στην υλοποίησή τους, το αποτέλεσμα είναι το ίδιο.

6 Παραδοχές

1. Δεν λαμβάνουμε υπόψιν την κατηγορία ταινίας “**unknown**”.
2. Λόγω του μεγάλου πλήθους των δεδομένων και της επεξεργασίας αυτών, ενδέχεται η εκτέλεση των αλγόριθμων να είναι λίγο χρονοβόρα.
3. Τα αρχεία των δεδομένων πρέπει να βρίσκονται στο ίδιο φάκελο με τα αρχεία της εργασίας. Ενδεικτικά παρουσιάζεται η δομή του project στο PyCharm.



Εικόνα 6.1