Assignment #2

by: Sirinian Aram Emmanouil AM: 2537

Ερώτηση 1η

Maximum Likelihood Estimation (MLE):

> Σε γενικές γραμμές, για ένα σταθερό σύνολο των δεδομένων και των υποκείμενων στατιστικών μοντέλων, η μέθοδος της μέγιστης πιθανοφάνειας επιλέγει το σύνολο των τιμών των παραμέτρων του μοντέλου που μεγιστοποιεί την συνάρτηση πιθανότητας.

Θέλουμε να βρούμε την παράμετρο θ=λ που θα μεγιστοποιεί την πιθανότητα $P(X|\theta)$

$$P(X|\theta)=L(\theta)=\prod_{i=1}^{n} P(xi|\theta)=\prod_{i=1}^{n} f(xi|\theta)$$

$$Llog(\theta)=\sum_{i=1}^{n} \log(P(xi|\theta))=\sum_{i=1}^{n} \log(f(xi|\theta))$$

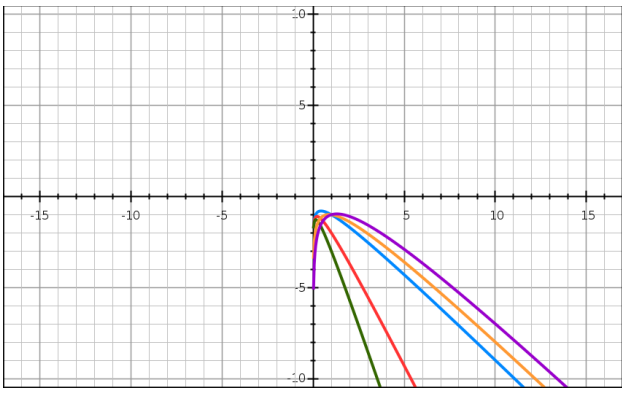
$$Llog(\lambda)=\sum_{i=1}^{n} [\log(\lambda)-\lambda\times xi]=n\times\log(\lambda)-\lambda\times\sum_{i=1}^{n} xi$$

$$\frac{d}{d\lambda} Llog(\lambda)=\frac{n}{\lambda}-\sum_{i=1}^{n} xi$$

Από το θεώρημα του Fermat συμπεραίνουμε ότι:

Οι πιθανές θέσεις των τοπικών ακροτάτων:

1) Τα εσωτερικά σημεία στα οποία η παράγωγος της f μηδενίζεται.
2) Τα εσωτερικά σημεία στα οποία η f δεν παραγωγίζεται.
3) Τα άκρα (αν ανήκουν στο πεδίο ορισμού της).



f(x) = log(x) -x, f(x) = log(x) -2x, f(x) = log(x) -3x, f(x) = 2log(x) -x, f(x) = 3log(x) -x

Από την πληροφορία του γραφήματος μπορούμε να αποκλείσουμε τα 2) και 3)

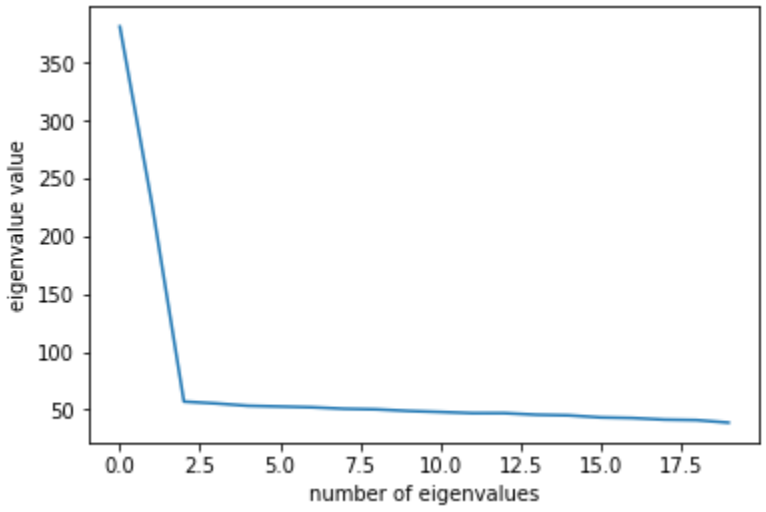
$$\frac{d}{d\lambda} Llog(\lambda)=0 \Leftrightarrow \frac{n}{\lambda}-\sum_{i=1}^{n} xi=0 \Leftrightarrow \frac{n}{\lambda}=\sum_{i=1}^{n} xi \Leftrightarrow \lambda=\frac{n}{\sum_{i=1}^{n} xi}$$

Η παράμετρο θ=λ που μεγιστοποιεί την πιθανότητα $P(X|\theta)$ είναι $\lambda=\frac{n}{\sum_{i=1}^{n} xi}$

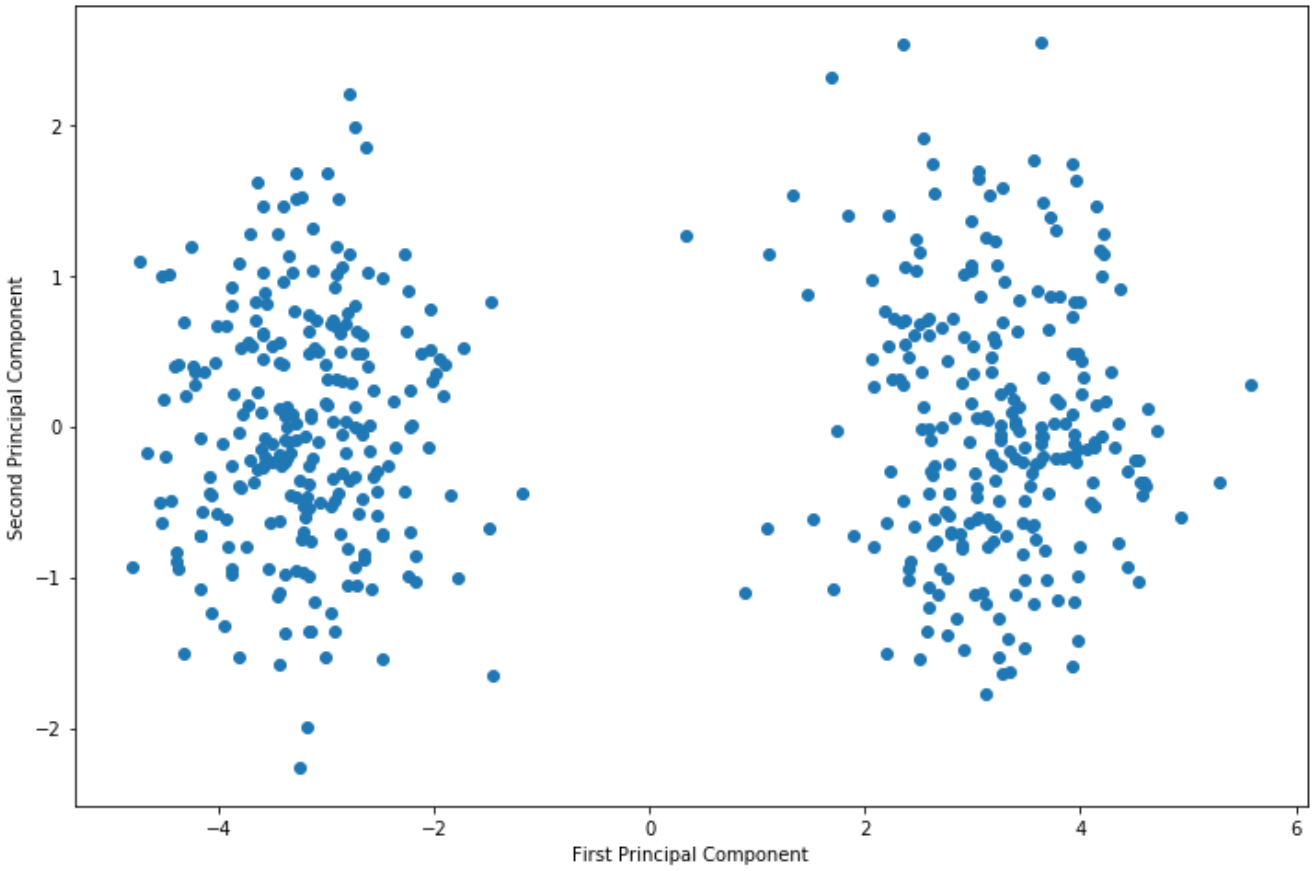
Ερώτηση 2η

Φορτώνοντας τα δεδομένα και εφαρμόζοντας τον αλγόριθμο Principal Component Analysis στη μία και στις δύο διαστάσεις. Εξετάζοντας τα δύο πρώτα components, παρατηρείτε ότι:

Είτε στην μία είτε στις δύο διαστάσεις παρατηρείτε ότι δημιουργούνται 2 ξεχωριστά clusters. Ο λόγος που τα διαγράμματα στη μια και στις δύο διαστάσεις δεν απέχουν πολύ μεταξύ τους είναι, ότι το ένα από τα components παίζει πολύ παραπάνω ρόλο στην περιγραφή των δεδομένων (εξήγηση των δεδομένων) σε σχέση με τα υπόλοιπα. Παρόλο αυτά και το δεύτερο είναι αρκετά σημαντικό επίσης. Η σημαντικότητα αυτήν υπολογίζεται από τις Eigenvalues τιμές όσο μεγαλύτερες τόσο ποιο σημαντικό το αντίστοιχο του attribute. Ένα ακόμα εύχρηστό εργαλείο για την εύρεση της “πραγματικής” διάστασης των δεδομένων είναι και τα corresponding singular values. Εμείς θεωρούμε ως κατάλληλο αριθμό από attributes αυτή που περιγράφει περίπου το 85% του variation. Με τα πρώτα δύο components έχουμε 83%.



Όσον αφορά τα δύο ξεχωριστά clusters που σχηματίζονται μπορούμε να συμπεράνουμε δυο πράγματα. Πρώτον ότι υπάρχουν δύο κατηγορίες από χρήστες στην δομή μας αυτοί που χρησιμοποιούν κυρίως τις νόμιμες ουσίες και αυτοί που χρησιμοποιούν κυρίως τις παράνομες. Το δεύτερο συμπέρασμα που μπορούμε να βγάλουμε είναι ότι αυτά τα σύνολα είναι πάνω κάτω του ίδιου μεγέθους, δηλαδή όσοι προμηθεύονται κυρίως νόμιμες ουσίες άλλοι τόσοι προμηθεύονται κυρίως παράνομες ουσίες.



Ερώτηση 3η

Από το ερώτημα 3 έχουν υλοποιηθεί όλα τα υποερωτήματα χωρίς όμως το bonus του υποερωτήματος 5.

Αρχικά γίνονται τα imports των κατάλληλων βιβλιοθηκών. Μετά πραγματοποιείτε η συλλογή των δεδομένων από τα αρχεία business.json και review.json. Χρησιμοποιώντας αυτά τα δεδομένα δημιουργείτε ένας πίνακας user-business(aka review_data) με τα rating των χρηστών για όλες τις επιχειρήσεις στην πόλη του “Toronto”. Ο πίνακας διαιρεί μόνο τους χρήστες που έχουν κάνει τουλάχιστον 10 ratings, και τις επιχειρήσεις που έχουν δεχτεί τουλάχιστον 10 ratings. Η διαδικασία αυτή γίνεται μέσα σε έναν επαναληπτικό βρόχο μέχρι όλοι οι χρήστες και όλες οι επιχειρήσεις στον πίνακα να έχουν τουλάχιστον 10 ratings.

Στην συνέχεια το πρόγραμμα αφαιρεί τυχαία ένα 10% των ratings. 10% (aka random_ps10) 90% (aka review_data)

Έπειτα γίνεται η κατασκευή των βασικών δομών από δεδομένα για να εκτελεστούν αργότερα οι κύριοι αλγόριθμοι. Η επιλογές των δομών αλλά και η επανάληψη των ίδιον δεδομένων μέσα σε διαφορετικές δομές πραγματοποιείτε για την αξιοποίηση των ιδιοτήτων τους σε ότι αφορά τον χρόνο εκτέλεσης και την “καθαρότητα” του κώδικα. Τέτοιες δομές είναι (review_data2_dict, user_data_g9(set), business_data_g9(set), user_data_g9(list), business_data_g9(list), review_data2, M(sparse matrix of review_data2)).

Με την χρήση του 90% των δεδομένων. Εκτελούνται οι αλγόριθμοι, User Average (UA), Business Average (BA), User-based Collaborative Filtering (UCF), Item-based Collaborative Filtering (ICF), Singular Value Decomposition (SVD).

Συνεχίζοντας μετά την κατασκευή των βασικών δομών, υλοποιούνται οι (UA) και (BA) με παρόμοιο τρόπο. Κατασκευάζονται οι δομές (averages_of_M(numpy.ndarray), averages_of_MT(numpy.ndarray)) που είναι πίνακες από μέσους όρους για κάθε business και user. Ύστερα δυο μέθοδοι getUA(user) και getBA(business) για εύκολη πρόσβαση στις τιμές.

Ακολουθεί η υλοποίηση του αλγόριθμου (UCF). Αυτό επιτυγχάνεται με την κατασκευή των μεθόδων (normalizedCosineSimilarity(array1, array2), mostSimilarUsers(user, business, k), equationUCF(user, business, k)). Για τον χρήστη u υπολογίζετε το σύνολο Nk(u) με τους k πιο όμοιους χρήστες οι οποίοι έχουν βαθμολογήσει την επιχείρηση b, αυτό επιτυγχάνεται με την mostSimilarUsers(user, business, k). Στη συνέχεια χρησιμοποιείτε η εξής εξίσωση για την πρόβλεψη:

$p(u,b)=\overline{r(u)}+\frac{\sum_{u'\in Nk(u)} s(u,u')(r(u',b)-\overline{r(u')})}{\sum_{u'\in Nk(u)} s(u,u')}$ με την equationUCF(user, business, k).

Για την ομοιότητα χρησιμοποιήστε το cosine similarity, μετά την αφαίρεση της μέσης τιμής από την κάθε γραμμή. Υλοποιείτε με την normalizedCosineSimilarity(array1, array2).

Μετά υλοποιείτε ο (ICF) αλγόριθμος. Αυτό επιτυγχάνεται με την κατασκευή των μεθόδων (cosineSimilarity(array1, array2), mostSimilarBusinesses(user, business, k), equationICF(user, business, k)). Για την επιχείρηση b υπολογίζετε το σύνολο Nk(b) με τις k πιο όμοιες επιχειρήσεις (σύμφωνα με το cosine similarity) οι οποίες έχουν βαθμολογηθεί από τον χρήστη u, αυτό επιτυγχάνεται με την mostSimilarBusinesses(user, business, k). Στη συνέχεια χρησιμοποιείτε η εξής εξίσωση για την πρόβλεψη: $p(u,b)=\frac{\sum_{b'\in Nk(m)} s(b,b')r(u,b')}{\sum_{b'\in Nk(b)} s(b,b')}$ με την equationICF(user, business, k).

Το cosine similarity υλοποιείτε με την cosineSimilarity(array1, array2).

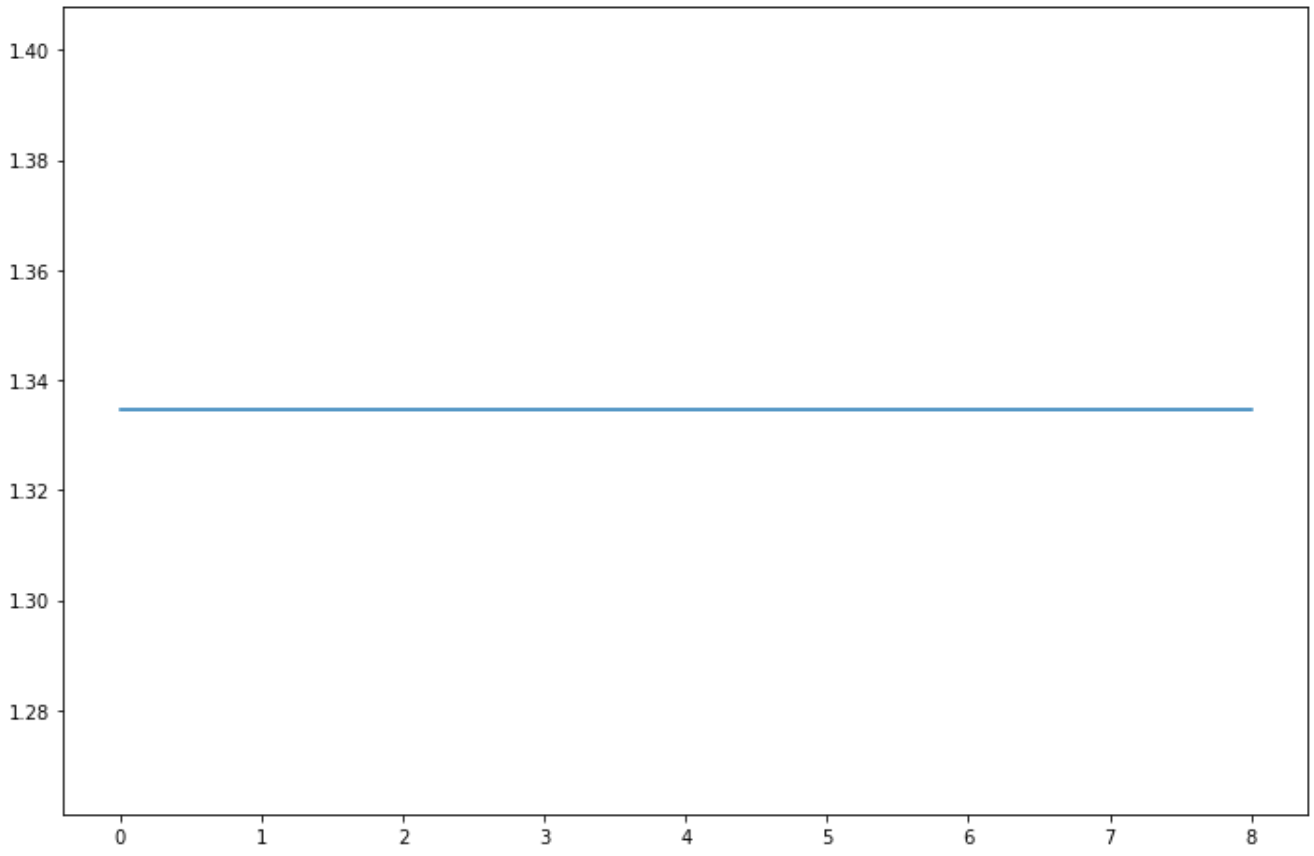
Ακολουθεί η υλοποίηση του αλγόριθμου (SVD). Αυτό επιτυγχάνεται με την κατασκευή των μεθόδων (plotSfromMatrixSVD(), matrixSVD(k), setMatrixSVD(k), equationSVD(user, business)). Αρχικά κατασκευάζονται οι πίνακες (U, s, Vh) με τις έτοιμες βιβλιοθήκες του numpy (np.linalg), οι οποίοι είναι απαραίτητοι για τον αλγόριθμο. Η συνάρτηση setMatrixSVD(k) καλεί την matrixSVD(k), η matrixSVD(k) χρησιμοποιώντας τα (U, s, Vh) μηδενίζει τα στοιχεία του s εκτός από τα πρώτα k (τα k μεγαλύτερα singular vectors) και επιστρέφει τον rank-k πίνακα Rk. Οι τιμές του k βρίσκονται αποθηκευμένες στον πίνακα k_values. Η απόφαση του k γίνεται με την παρακολούθηση των singular τιμών για να αποφασίσετε το μέγεθος του k, η παρακολούθηση του ευκολύνετε με την συνάρτηση plotSfromMatrixSVD() η οποία σχεδιάζει μια γραφική παράσταση με τις singular vectors τιμές. Η συνάρτηση equationSVD(user, business) επιστρέφει την τιμή στον πίνακα matrix_svd για το ζευγάρι (user, business).

Ακολουθεί ο αλγόριθμος Root Mean Square Error (RMSE), για την αξιολόγηση και σύγκριση των αλγορίθμων. Αν r1 , r2 , ... , r n είναι τα ratings που θέλουμε να προβλέψουμε, και p1 , p2 , ... , pn είναι οι προβλέψεις του αλγορίθμου, το RMSE του αλγορίθμου ορίζεται ως $RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(ri-pi)^2}$.

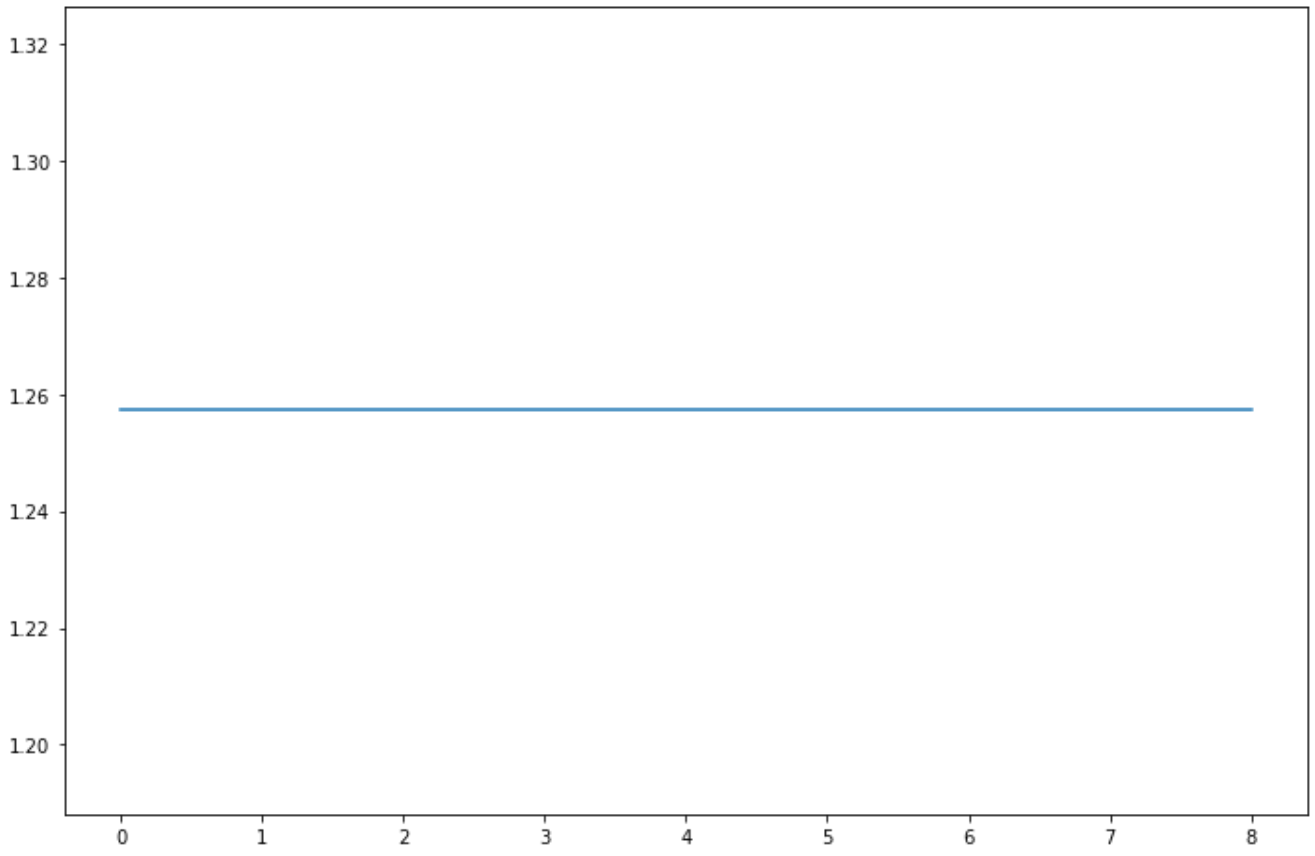
Αυτό επιτυνχάνεται με την κατασκευεί των μεθόδων (calculateRMSE(array_r, array_p)). Η calculateRMSE(array_r, array_p) παίρνει σαν είσοδο μια λίστα από τα ratings που θέλουμε να προβλέψουμε και array_p από το σύνολο των 10% που αφαιρέσαμε (aka random_ps10).

Για την σύγκριση των αλγορίθμων υπάρχουν δύο μεταβλητές ο number_of_ratings και ο number_of_k_values, η τιμή των οποίων επηρεάζουν σημαντικά την ταχύτητα εκτέλεσης του προγράμματος. Για τα παρακάτω αποτελέσματα οι τιμές είναι (number_of_ratings = 10, number_of_k_values = 9). Ο κάθετος άξονας μετράει την τιμή της ομοιότητας, ο επίπεδος άξονας μετράει την τιμή του k (number_of_k_values).

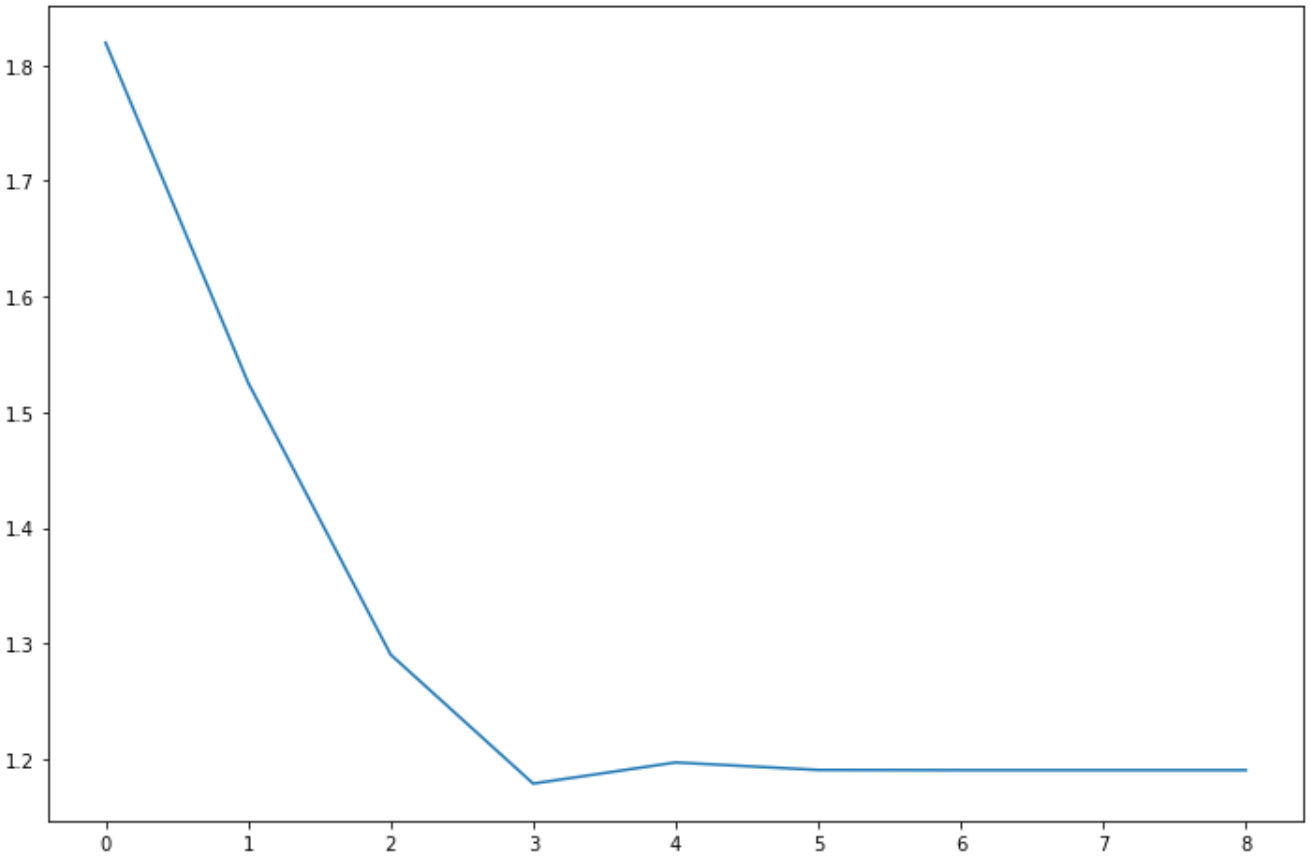
Για τον αλγόριθμο (UA) το k δεν επηρεάζει τα αποτελέσματα μιας και δεν παίζει ρόλο στην παραγωγή των αποτελεσμάτων για’τό και στην γραφική παράσταση παίρνουμε μια επίπεδη σταθερή γραμμή στην τιμή (1.33).



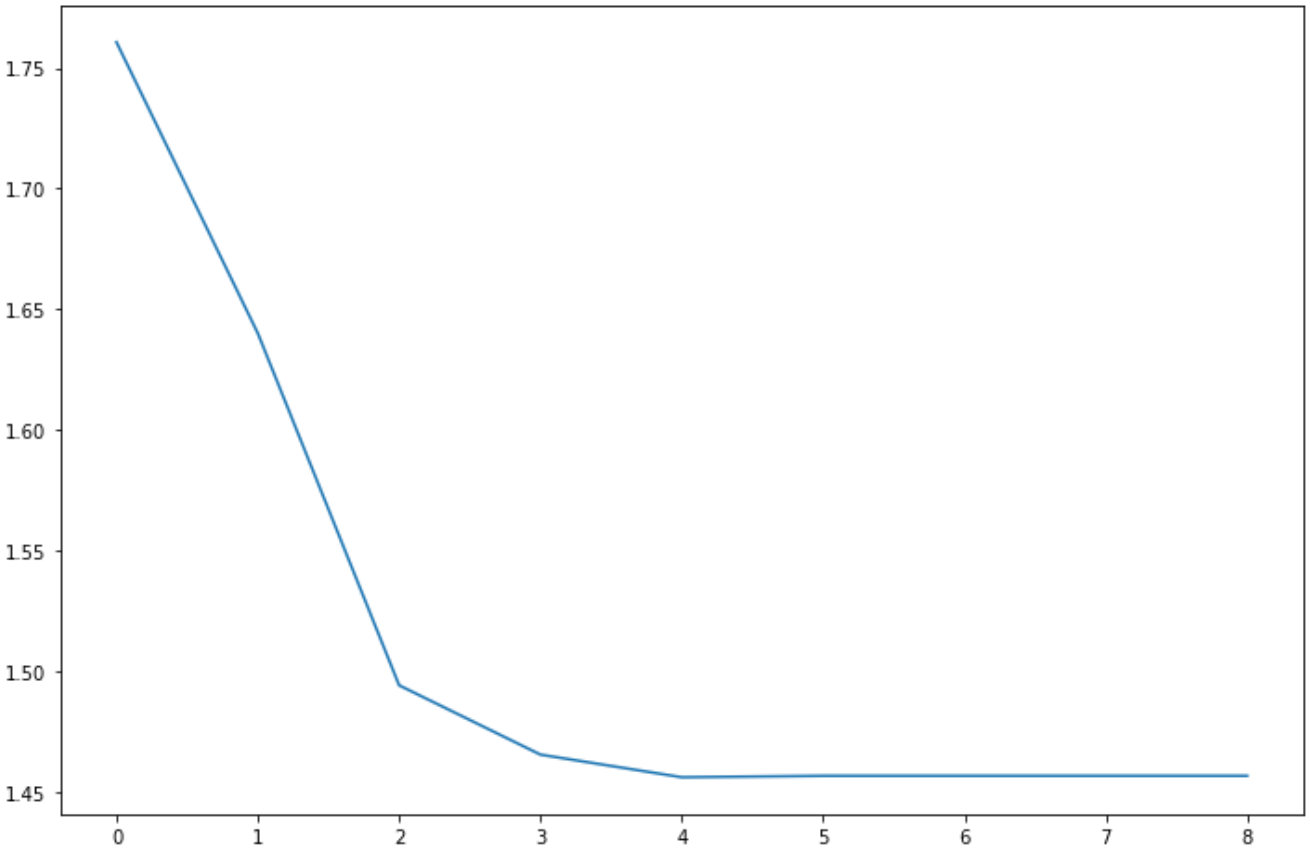
Για τον αλγόριθμο (BA) το k δεν επηρεάζει τα αποτελέσματα μιας και δεν παίζει ρόλο στην παραγωγή των αποτελεσμάτων για’υτό και στην γραφική παράσταση παίρνουμε μια επίπεδη σταθερή γραμμή στην τιμή (1.26).



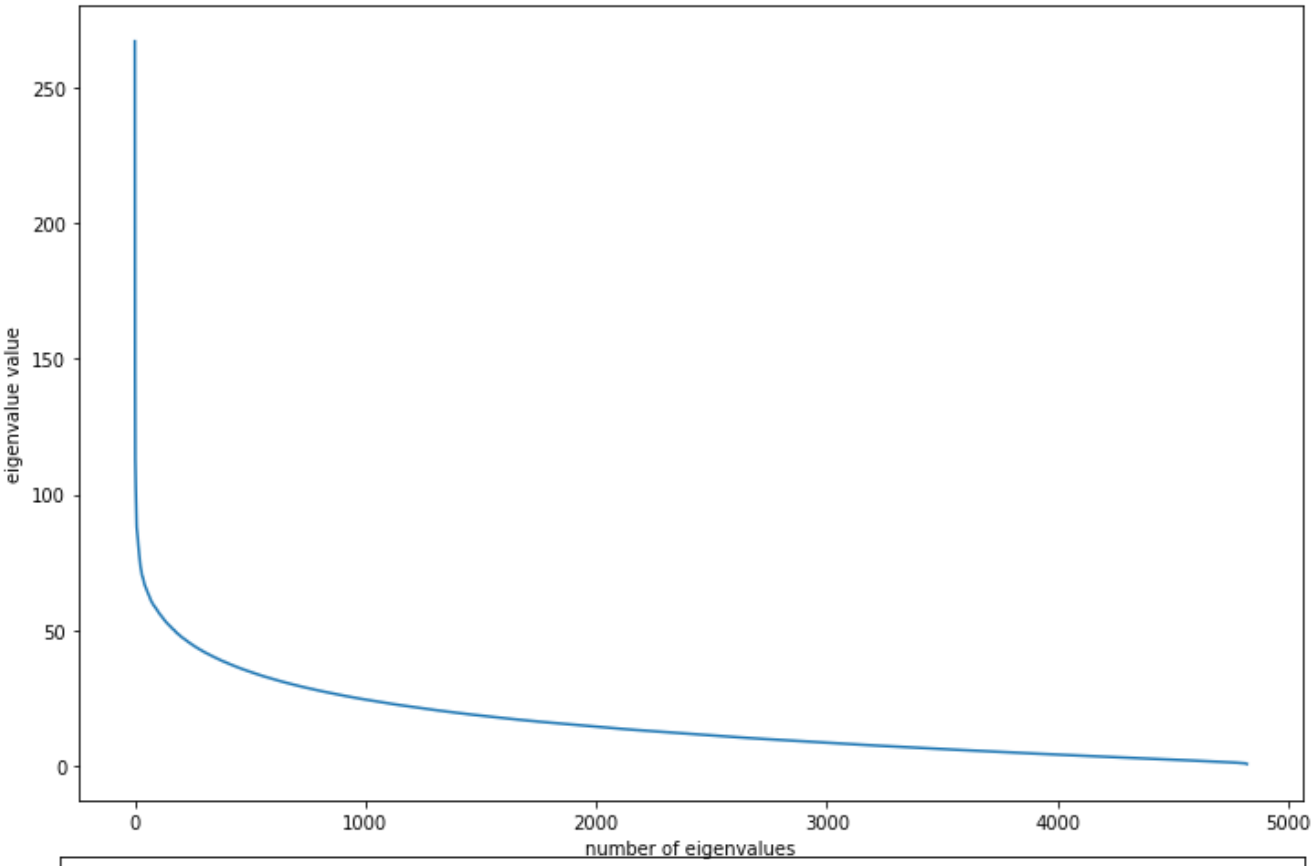
Για τον αλγόριθμο (UCF) το k επηρεάζει τα αποτελέσματα μιας και καθορίζουν το μέγεθος του Nk(u) συνόλου με τους k πιο όμοιους χρήστες (σύμφωνα με το correlation coefficient) οι οποίοι έχουν βαθμολογήσει την επιχείρηση b. Αυτό που παρατηρούμε από την γραφική παράσταση είναι ότι οι καλύτερες προσεγγίσεις γίνονται όταν το k παίρνει τιμές μέσα στο (2, 8] σύνολο.

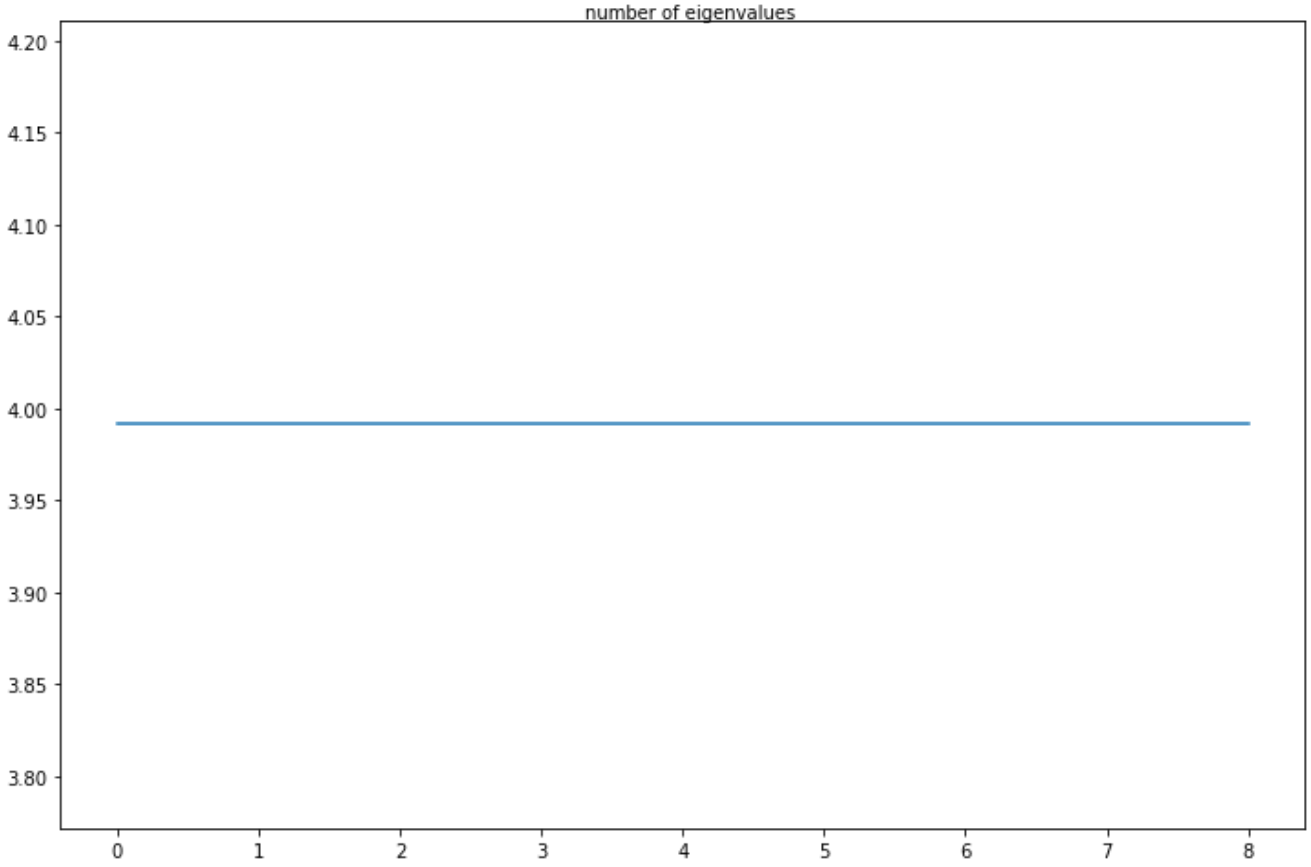


Για τον αλγόριθμο (ICF) το k επηρεάζει τα αποτελέσματα μιας και καθορίζουν το μέγεθος του Nk(b) με τις k πιο όμοιες επιχειρήσεις (σύμφωνα με το cosine similarity) οι οποίες έχουν βαθμολογηθεί από τον χρήστη u. Αυτό που παρατηρούμε από την γραφική παράσταση είναι ότι οι καλύτερες προσεγγίσεις γίνονται όταν το k παίρνει τιμές μέσα στο (2, 8] σύνολο.



Για τον αλγόριθμο (SVD) το k καθορίζει πόσα από τα μεγαλύτερα singular vectors θα κρατήσει για να κατασκευάζει τον rank-k πίνακα Rk. Από το πρώτο διάγραμμα μπορούμε να αποφανθούμε ότι οι καλύτερες τιμές είναι στο γόνατο του διαγράμματος περίπου (0, 1000). Κανονικά αυτό θα έπρεπε να αντικατοπτριστεί και στην δεύτερη παράσταση κάτι που δεν συμβαίνει και για’υτό παραμένει ανοιχτό ερώτημα ο λόγος ή το λάθος που το προκαλεί.





Ερώτηση 4η

Από αυτό το ερώτημα δεν έχουν υλοποιηθεί τα προβλήματα παρά μόνο η εξαγωγή των χρήσιμων δεδομένων. Ποιο συγκεκριμένα οι επιχειρήσεις από το Τορόντο, αυτές που βρίσκονται σε γειτονιές με τουλάχιστον 300 επιχειρήσεις.