ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# Project

Μέρος 3ο

**Όνομα**
Κυλάφη Χριστίνα-Θεανώ

**ΑΜ**
1115201200077

**Αθήνα, 2019**

Η εργασία έχει υλοποιηθεί σε **γλώσσα C**, στο πρόγραμμα "**sublime text**" ενώ παράλληλα δοκιμαζόταν σε περιβάλλον **linux** σε Virtual Machine( **VirtualBox** ), όπως και απομακρυσμένα σε μηχάνημα linux και γινόταν υποβολή (**push**) με τη χρήση εντολών **git** στο **bitbucket**, σε **private repository**.

Στο φάκελο, περιλαμβάνονται τα εξής αρχεία εκτός από το **readme** :
**makefile**, **extras.c**, **extras.h**, **recom_steps.c**, **recom_steps.h**, **recom_structs.c**, **recom_structs_h**, **recom.c**, **recom.h**, **recommend.c**, **recommend.h**, **validation.c**, **validation.h**, **lsh.c**, **lsh.h**, **cube.c**, **cube.h, cubefuns.c**, **cubefuns.h**, **hash.c**, **hash.h**, **structfuns.c**, **structs.h**, **extrafuns.c**, **extrafuns.h**, **cluster.c**, **cluster.h**, **steps_of_clustering.c**, **steps_of_clustering.h**, **initialization.c**, **initialization.h**, **assignment.c**, **assignment.h**, **update.c**, **update.h**, **cluster.conf**, **readme_part1/part2** (τα readme του 1ου και 2ου μέρους της εργασίας, που αφορούν αλγορίθμους που αξιοποιούμε στο part 3 (LSH / Clustering) και τα δοσμένα αρχεία input/λεξικά κλπ.

Με την εντολή "**make**" παράγεται το εκτελέσιμο "**recommendation**".

Η σύνταξη για την εκτέλεση του παραπάνω, είναι:
**./recommendation –d** <input file> **-o** <output file> ( **recommendation** algorithm )
**./recommendation –d** <input file> **-o** <output file> **-validate** ( **validation** of algorithms )

**>** Λίγες λεπτομέρειες για τα αρχεία:

— **recom.c:** Η main συνάρτηση του προγράμματος, στην οποία αποθηκεύονται τα δεδομένα (λεξικά όρων, συναισθημάτων, tweets, κλπ), και ανάλογα με τα ορίσματα, "τρέχει" είτε τον αλγόριθμο πρόβλεψης (recommendation), είτε την αξιολόγηση των μεθόδων(validation).

— **recom_steps.c:** Βήμα βήμα το Recommendation , δηλαδή η **προετοιμασία** των δεδομένων, η κατάλληλη **επεξεργασία** και έπειτα οι βασικοί αλγόριθμοι με βάση τους οποίους προκύπτουν τα αποτελέσματα. Συγκεκριμένα, υπολογίζεται το **σκορ** των tweets ( **calculate_sentim_score( )** ), δημιουργούνται οι **χρήστες**( **create_users( )** ) που περιέχουν σαν δομή, εκτός των άλλων, και ένα σύνολο από **tweets** και ένα διάνυσμα με συναισθηματικές **αξιολογήσεις** των **κρυπτονομισμάτων** που περιέχουν τα tweets τους (πρόκειται για το διάνυσμα με το οποίο αργότερα θα γίνουν τα updates των άγνωστων αξιολογήσεων καθώς και οι τελικές προτάσεις των καλύτερων προβλέψεων ). Έπειτα, υπολογίζεται ο **μέσος όρος** ( **calculate_average_rating( )** ) και αποθηκεύεται η αρχική αυτή κατάσταση του διανύσματος σε ένα 2ο διανυσμα στο χρήστη ( **user_crypto_save_users( )** ). Αφαιρώ τους μηδενικούς χρήστες από το dataset( **clear_users_with_no_coins( )** ), γιατί δε θα συμβάλλουν τελικά στις προβλέψεις, και δημιουργώ τους **εικονικούς** χρήστες, με το **clustering** method της εργασίας 2( **clustering( )** ), για τους οποίους και πάλι βρίσκω το μέσο όρο και αποθηκεύω την αρχική κατάσταση του διανύσματος των αξιολογήσεων των κρυπτονομισμάτων.
Έπειτα, εκτελώ τους **αλγορίθμους πρόβλεψης**, με ενδιάμεσα "**resets**" / **καθαρισμούς** των χρηστών( **retrieve_saved_user_crypto_ratings_users( )** / **clear_users( )** ), δηλαδή **επαναφορά** κάθε δομής στην **αρχική της κατάσταση**, δηλαδή την κατάσταση ακριβώς πριν την εκτέλεση του πρώτου αλγορίθμου. **Γράφω** στο **αρχείο** τις καλύτερες προβλέψεις που έχουν προκύψει, **εκτυπώνω** στο **τερματικό** την ίδια λίστα, **απελευθερώνω** τη **μνήμη** που είχαν χρησιμοποιήσει οι δομές και **τερματίζει** το πρόγραμμα (**recommendation**).

— r**ecommend.c:** Εδώ υλοποιούνται οι βασικές συναρτήσεις/αλγόριθμοι/διαδικασίες του recommendation.
Όσον αφορά τη συνάρτηση **cosine_LSH_recom_1_2( )**, πρόκειται για τον αλγόριθμο **LSH** και τους **P κοντινότερους** γείτονες κάθε user, μέσα από τους οποίους **υπολογίζονται** οι **άγνωστες αξιολογήσεις** βάσει του τύπου στις διαφάνειες. Αρχικά δημιουργείται μια **λίστα** με τους users( **struct multipoints*** ), δημιουργούνται τα **hashtables**( **lsh_range( )** ) με βάση αυτή τη

λίστα( δομή από εργασία 1η ) και τελικά γίνεται η **αναζήτηση** των P κοντινότερων γειτόνων του κάθε user από τη λίστα ( **lsh_p_range( )** ), κάνοντας κανονικά τη διαδικασία που κάναμε και στην πρώτη εργασία ( hashing του user, εύρεση των hashtables στα οποία ανήκει, συγκρίσεις αποστάσεων και παραγωγή λίστας με τους P κοντινότερους γείτονες - **LSH** algorithm ).
Αφού λοιπόν διαμορφώθηκαν οι άγνωστες τιμές στα διανύσματα των users, με τη συνάρτηση **find_best( )** βρίσκουμε τα καλύτερα ( 5 ή 2 ανάλογα με το ζητούμενο στο οποίο βρισκόμαστε ), και τα αποθηκεύουμε στις κατάλληλες δομές μέσα στους χρήστες ( **best_5_real** / **best_2_represent** ). Διαγράφουμε τις δομές που χρησιμοποιήθηκαν και πλέον δε χρειαζόμαστε, και επιστρέφει η συνάρτηση.
Όμοια και για την **cosine_LSH_recom_1_2**, μόνο που αντί να δημιουργήσω τα **hashtables** με τη λίστα των users, τώρα θα έχω τη λίστα των **virtual users**, ώστε κάθε ένας πραγματικός χρήστης, με τον LSH **να βρεθεί σε buckets εικονικών χρηστών**, και από εκείνους να βρούμε τους P κοντινότερους.
Επόμενη μέθοδος / συνάρτηση είναι η **clustering_recom_2_1( )**, με την οποία εκτελείται το clustering recommendation, με τους **πραγματικούς** χρήστες. Δημιουργείται όπως και πριν μια **λίστα** με τους users, γίνεται **clustering** αυτών των χρηστών, παράγονται clusters χρηστών και στη συνέχεια, καταγράφεται σε ποιο cluster ανήκει ο κάθε χρήστης. Για κάθε χρήστη λοιπόν, δημιουργείται μια λίστα τύπου **struct simple_list***, για να αποτελέσει είσοδο στη συνάρτηση **find_best( )** όπως και στους προηγούμενους αλγορίθμους, και να υπολογιστούν από τους **users του cluster αυτού και μόνο,** οι **προβλέψεις** των **άγνωστων** αξιολογήσεων του εκάστοτε χρήστη. Συμπληρώνεται και πάλι η δομή με τις καλύτερες προτάσεις, διαγράφονται οι δομές που χρησιμοποιήθηκαν και επιστρέφει η συνάρτηση.
Τέλος, η **clustering_recom_2_2( )** συνάρτηση, είναι όμοια με την ακριβώς από πάνω, μόνο που σε αυτή την περίπτωση, κάνουμε clustering το σύνολο των **εικονικών χρηστών με κάθε χρήστη**, **έναν- έναν**. Δημιουργούμε δηλαδή μια λίστα με virtual users και μια με τους πραγματικούς, και διατρέχοντας τη λίστα με τους πραγματικούς, σε κάθε επανάληψη προσθέτουμε τον εκάστοτε πραγματικό χρήστη στη λίστα με τους εικονικούς. Εφαρμόζουμε λοιπόν τα παραπάνω βήματα, σε αυτή τη λίστα, καταλήγοντας λοιπόν να **υπολογίζουμε** τα **άγνωστα** ratings από τους **εικονικούς** χρήστες - **γείτονες** του κάθε πραγματικού χρήστη, **στο ίδιο cluster**.

— **recom_structs.c:** Εδώ υλοποιούνται οι δομές και οι συναρτήσεις χειρισμού τους, που χρησιμοποιούνται στους παραπάνω αλγορίθμους, δηλαδή initialization, διαγραφές, εκτυπώσεις, προσθήκη κόμβου, διαγραφή, get, save, κλπ

— **validation.c:** Σε αυτό το αρχείο είναι συγκεντρωμένες όλες οι διαδικασίες και δομές σχετικές με την αξιολόγηση των αλγορίθμων. Συγκεκριμένα, η **αξιολόγηση**, υλοποιείται ως εξής. Δημιουργώ κανονικά τους **χρήστες** (εικονικούς και πραγματικούς) όπως και στο recommendation μέρος του προγράμματος, **αποθηκεύω** τις **γνωστές** τιμές των **ratings** για τα κρυπτονομίσματα σε πίνακα ( **struct known_ratings*** ), “σπάω” τη λίστα σε 10 μέρη-**10 fold cross validation** ( **struct validation_set*** ), ώστε να χρησιμοποιώ το κάθε μέρος σε κάθε επανάληψη ως **validation set** και εκτελώ την αξιολόγηση, την οποία έχω διαχωρίσει σε 2 μέρη ( **validate_A( )** , **validate_B( )** ), ένα για κάθε ζητούμενο. Στις επιμέρους αυτές συναρτήσεις, εκτελώ σε επανάληψη τους recommendation αλγορίθμους που περιέγραψα παραπάνω, μόνο που κάθε φορά, πριν από την εκτέλεση, με τη συνάρτηση **known_to_unknown( ) μετατρέπω** τα **γνωστά** ratings του validation set ( κάθε rating δηλαδή ενός συγκεκριμένου user, για ένα συγκεκριμένο κρυπτονόμισμα ) σε άγνωστα, **αντικαθιστώντας** την τιμή του με την τιμή που έχουν τα άγνωστα ( “inf” ). Πριν από κάθε επόμενη επανάληψη, **υπολογίζω** το **απόλυτο σφάλμα** του υποσυνόλου σύμφωνα με τους τύπους από τις διαφάνειες, **κρατώ** τις τιμές από όλες τις επαναλήψεις για τον τελικό υπολογισμό του **μέσου**, πια, **απολύτου σφάλματος**, **επαναφέρω** τις γνωστές αρχικές τιμές των ratings ( με την **unknown_to_known( )** ), και συνεχίζω με το επόμενο υποσύνολο γνωστών ratings, ως validation set, αφού **καθαρίσω** όπως και προηγουμένως, τη δομή **users**.

— **extras.c:** Βοηθητικές, απλές συναρτήσεις που χρειάστηκαν (αντίστοιχες με **extrafuns.c** της 2ης εργασίας)

Το πρόγραμμα έχει ως εξής (**γενική ιδέα**) :

- Έχοντας τα δοσμένα αρχεία με τα **sentiments** των λέξεων που περιέχονται στα tweets, τα **tweets** και τα **κρυπτονομίσματα**, **αποθηκεύω** τα δεδομένα σε κατάλληλες **δομές** που έχω σχεδιάσει.
- Δημιουργώ τους **users**, που περιέχουν ένα **σύνολο** από **tweets** που έχουν παράξει, ο καθένας, τα οποία tweets, χαρακτηρίζονται από ένα **σκορ**
- Αυτοί λοιπόν οι **users** θα περάσουν μια **επεξεργασία** ώστε να καταλήξουμε να έχουμε για τα **unrated** coins τους, **προβλέψεις** τιμής.
- Η επεξεργασία είναι είτε ο αλγόριθμος **LSH** ώστε να κάνουμε τις προβλέψεις, από τους P κοντινότερους γείτονες του κάθε user μέσα από τα buckets των hashtables στα οποία "πέφτει", είτε **clustering** ώστε να υπολογίσουμε τις προβλέψεις από τις τιμές των users στο ίδιο **cluster** που ανήκει και ο εκάστοτε user ( τα παραπάνω γίνονται τη μια φορά με τους πραγματικούς users - A και την άλλη με τους εικονικούς - B )
- Ύστερα από την παραπάνω επεξεργασία, έχουμε **5** προβλέψεις που προέκυψαν από τη διαδικασία με τους **πραγματικούς** χρήστες και **2**, από τη διαδικασία με τους **εικονικούς**
- **Εκτυπώνονται** τα αποτελέσματα στα αρχεία και απελευθερώνεται η μνήμη
- Στην περίπτωση του **validation**, γίνεται ό,τι και παραπάνω, μόνο που σε αυτή την περίπτωση, **χωρίζουμε** τα **γνωστά** ratings σε 10 μέρη (**10 fold cross validation**), ώστε να πάρουμε κάθε φορά το ένα μόνο μέρος ως **validation set**, κάνοντας άγνωστα τα συγκεκριμένα ratings.
- Έπειτα, όπως ακριβώς στον υπολογισμό κάθε **σφάλματος**, κάνουμε τις απαραίτητες πράξεις και προκύπτει το **μέσο απόλυτο σφάλμα** των 2 μεθόδων για κάθε διαδικασία( πραγματικοί/ εικονικοί χρήστες - **LSH**/**Clustering** )

## > Κύριες δομές :

( οι παρακάτω δομές, χρησιμοποιήθηκαν ως βάση για την παραγωγή περαιτέρω μεγαλύτερων δομών, όπως λίστες με αυτούς τους κόμβους, δυναμικών πινάκων, κλπ )

**—> struct users:** Η κύρια δομή που αποτελεί έναν χρήστη, δηλαδή ένα σύνολο από tweets, στοιχεία και πληροφορίες όπως το **id** του, πόσα **tweets** περιέχει, σε πόσα **νομίσματα** αναφέρεται, τα **ratings** του κάθε νομίσματος, τα **καλύτερα** νομίσματα, σύμφωνα με τις προβλέψεις από κάθε ζητούμενο, κλπ.

**—> struct tweets:** Η δομή που αντιστοιχεί στα tweets

**—> struct sentim_score_map:** Δομή **αποθήκευσης** των **συναισθημάτων** της κάθε **λέξης**, για μετέπειτα χρήση στον **υπολογισμό** του **score** του κάθε tweet.

**—> struct crypto_coins:** Δομή **αποθήκευσης** των **κρυπτονομισμάτων**, κάθε κόμβος της οποίας περιέρχει μια **λίστα με ονομασίες/αναφορές**, που αντιστοιχούν στο νόμισμα του οποίου το **ολοκληρομένο/επίσημο όνομα**, είτε βρίσκεται στην **5η θέση** της λίστας, είτε στην **1η**, αν δεν περιέχει 5 αναφορές.

**—> known_ratings:** Δομή με όλα τα απαραίτητα στοιχεία για την αποθήκευση των γνωστών ratings του dataset. Αφού δημιουργήσουμε τους users με τα tweets και τα coins στα οποία αναφέρονται, διατρέχουμε το διάνυσμα crypto-user αξιολογήσεων, και **καταγράφουμε** τον user, το coin και το rating του συγκεκριμένου ζεύγους. Κατά το διαχωρισμό του set αυτού σε subsets , χρησιμοποιούμε και το **node_index**, για την **τυχαία επιλογή** αυτών των **ζευγών**, σε κάθε **subset**.

**—> validation_set:** Αποτελεί τη δομή που **αποθηκεύει** τα 10 **υποσύνολα**/validation subsets, και συγκεκριμένα, το σύνολο των ζευγών που περιέχει ακριβώς, καθώς και μια **λίστα από indexes**(από τον πίνακα των ζευγών με τα γνωστά ratings) των **ζευγών** που **περιέχει**.

## > Παρατηρήσεις :

**—>** Έχει γίνει χρήση του πρώτου και δεύτερου μέρους της εργασίας ( **LSH / Clustering**). Τροποποιήθηκε κατάλληλα, ώστε να ικανοποιήσει τις απαιτήσεις του τωρινού μέρους, δηλαδή το **Recommendation**, στα βήματα της δημιουργίας εικονικών χρηστών, καθώς και στους κύριους αλγορίθμους πρόβλεψης των unrated coins σε κάθε χρήστη.

**—>** Έχει χρησιμοποιηθεί γενικά μεγάλος βαθμός **κατακερματισμού** των συναρτήσεων στο πρόγραμμα, ώστε να είναι προσαρμόσιμο και εύκολα συντηρήσιμο. Κάθε κύρια συνάρτηση, “σπάει” σε μικρότερες και πιο βασικές, για την εύκολη **επαναχρησιμοποίηση** και διόρθωση/ **έλεγχό** τους.

**—>** Όπως αναφέρεται και στην εκφώνηση, το ζητούμενο για την υλοποίηση του οποίου χρησιμοποιείται ο αλγόριθμος **LSH**, χρησιμοποιεί την **cosine** μετρική, ενώ αντίθετα στον αλγόριθμο **συσταδοποίησης**, χρησιμοποιούμε την **ευκλείδεια** μετρική.

## > Ενδεικτικά Αποτελέσματα

Cosine LSH Recommendation —>

```
> Cosine LSH Recommendation
<u11>    none   ---||---   none
<u15>    none   ---||---   mixin   ripple
<u24>    ethereum   ---||---   ethereum   mixin
<u25>    ethereum   ---||---   bitcoin   ethereum
<u30>    ark   bitcoin cash   neo   pundi x   omisego   ---||---   mixin   bitcoin gold
<u36>    crypto.com   bitcoin cash   neo   pundi x   omisego   ---||---   ethereum   ripple
<u37>    bitcoin cash   bitcoin cash   neo   pundi x   omisego   ---||---   chainlink   litecoin
<u39>    bitcoin cash   bitcoin cash   neo   pundi x   omisego   ---||---   chainlink   litecoin
<u40>    bitcoin cash   bitcoin cash   neo   pundi x   omisego   ---||---   bitcoin   ethereum
<u43>    litecoin   ripple   bitcoin cash   pundi x   omisego   ---||---   mixin   ripple
<u50>    litecoin   ripple   bitcoin cash   pundi x   omisego   ---||---   mixin   vechain
<u53>    cybermiles   steem   digixdao   dogecoin   funfair   ---||---   tezos   maidsafecoin
<u62>    cybermiles   steem   digixdao   dogecoin   funfair   ---||---   ethereum   ripple
<u66>    crypto.com   steem   digixdao   dogecoin   funfair   ---||---   chainlink   ripple
<u71>    bitcoin cash   steem   digixdao   dogecoin   funfair   ---||---   mixin   cardano
<u77>    bitcoin cash   steem   digixdao   dogecoin   funfair   ---||---   mixin   cardano
<u78>    bitcoin cash   steem   digixdao   dogecoin   funfair   ---||---   mixin   gas
<u81>    bitcoin cash   steem   digixdao   dogecoin   funfair   ---||---   bitcoin   ripple
<u86>    bitcoin cash   ark   tenx   stellar   tron   ---||---   ethereum   ripple
<u87>    bitcoin cash   ark   tenx   stellar   tron   ---||---   ethereum   ripple
<u92>    ethereum   bitcoin cash   omisego   eos   neo   ---||---   ethereum   tezos
<u94>    cardano   zcoin   horizen   pivx   funfair   ---||---   ethereum   tezos
<u95>    verge   stellar   digibyte   pivx   funfair   ---||---   ethereum   tezos
<u100>   monacoin   ripple   dropil   tether   funfair   ---||---   ethereum   tezos
<u101>   monacoin   ripple   dropil   tether   funfair   ---||---   ethereum   ripple
<u103>   monacoin   ripple   dropil   tether   funfair   ---||---   ethereum   ripple
<u107>   nebulas   iostoken   maker   horizen   maidsafecoin   ---||---   vechain   mixin
<u119>   nebulas   iostoken   maker   horizen   maidsafecoin   ---||---   ethereum   mixin
<u120>   ark   tenx   digibyte   bitcoin cash   maidsafecoin   ---||---   horizen   chainlink
<u125>   ark   tenx   digibyte   bitcoin cash   maidsafecoin   ---||---   ethereum   ripple
<u128>   litecoin   bitcoin cash   digibyte   bitcoin cash   maidsafecoin   ---||---   chainlink   litecoin
<u132>   litecoin   bitcoin cash   digibyte   bitcoin cash   maidsafecoin   ---||---   ethereum   ripple
<u147>   litecoin   bitcoin cash   digibyte   bitcoin cash   maidsafecoin   ---||---   ethereum   mixin
<u148>   litecoin   bitcoin cash   digibyte   bitcoin cash   maidsafecoin   ---||---   ethereum   ripple
<u152>   litecoin   bitcoin cash   digibyte   bitcoin cash   maidsafecoin   ---||---   mixin   vechain
<u154>   litecoin   bitcoin cash   digibyte   bitcoin cash   maidsafecoin   ---||---   bitcoin   ripple
<u159>   ark   tenx   cardano   digibyte   maidsafecoin   ---||---   komodo   chainlink
<u166>   ark   tenx   cardano   digibyte   maidsafecoin   ---||---   komodo   chainlink
<u172>   ark   tenx   cardano   digibyte   maidsafecoin   ---||---   komodo   chainlink
<u186>   nebulas   iostoken   horizen   digibyte   dragonchain   ---||---   horizen   nexus
<u192>   wax   tenx   funfair   dogecoin   eos   ---||---   tezos   maidsafecoin
<u201>   tether   monacoin   tenx   ontology   ethereum classic   ---||---   tezos   maidsafecoin
<u205>   tether   monacoin   tenx   ontology   ethereum classic   ---||---   tezos   maidsafecoin
<u209>   tether   monacoin   tenx   ontology   ethereum classic   ---||---   tezos   maidsafecoin
<u211>   tether   monacoin   tenx   ontology   ethereum classic   ---||---   mixin   ripple
<u214>   ethereum   litecoin   ripple   bitcoin cash   ethereum classic   ---||---   ethereum   litecoin
<u218>   ethereum   tenx   litecoin   eos   monero   ---||---   verge   basic attention token
<u219>   ethereum   tenx   litecoin   eos   monero   ---||---   bitcoin   ethereum
<u221>   ethereum   tenx   litecoin   eos   monero   ---||---   bitcoin   ethereum
<u222>   ethereum   tenx   litecoin   eos   monero   ---||---   mixin   ripple
<u224>   tenx   waltonchain   huobi token   pundi x   eos   ---||---   waves   funfair
<u226>   tenx   waltonchain   huobi token   pundi x   eos   ---||---   ethereum   ripple
```

Clustering <—Recommendation

```
> Clustering Recommendation
<u11>    ripple   iota   stellar   dogecoin   dash   ---||---   bitcoin   ethereum
<u15>    crypto.com   neo   verge   tenx   funfair   ---||---   nano   chainlink
<u24>    crypto.com   neo   verge   tenx   funfair   ---||---   horizen   chainlink
<u25>    monacoin   bitcoin private   mixin   cryptonex   eos   ---||---   ripple   ethereum
<u30>    neo   omisego   digixdao   bitcoin cash   ark   ---||---   chainlink   tenx
<u36>    nano   cybermiles   tether   bancor   aeternity   ---||---   ripple   tezos
<u37>    verge   bitcoin cash   tether   holo   eos   ---||---   cardano   holo
<u39>    ethereum   litecoin   dash   eos   bytecoin   ---||---   ethereum   kyber network
<u40>    ethereum   litecoin   dash   eos   bytecoin   ---||---   tenx   eos
<u43>    iota   electroneum   vechain   pundi x   omisego   ---||---   waves   cardano
<u50>    icon   digibyte   vechain   pundi x   omisego   ---||---   tenx   horizen
<u53>    ark   bitcoin cash   digixdao   cybermiles   eos   ---||---   holo   funfair
<u62>    ripple   vechain   verge   waves   golem   ---||---   ethereum   ripple
<u66>    icon   dash   digibyte   chainlink   golem   ---||---   ripple   nxt
<u71>    digixdao   ardor   mixin   reddcoin   monero   ---||---   monero   holo
<u77>    digixdao   ardor   mixin   reddcoin   monero   ---||---   verge   ripple
<u78>    tenx   ardor   mixin   reddcoin   monero   ---||---   chainlink   horizen
<u81>    cybermiles   tether   bancor   nano   aeternity   ---||---   tenx   horizen
<u86>    tenx   litecoin   ripple   cardano   aeternity   ---||---   ethereum   nano
<u87>    tether   litecoin   ripple   cardano   aeternity   ---||---   lisk   verge
<u92>    cybermiles   dentacoin   tron   zcoin   icon   ---||---   lisk   verge
<u94>    cryptonex   nexus   komodo   iostoken   aeternity   ---||---   kyber network   elastos
<u95>    cryptonex   nexus   komodo   iostoken   aeternity   ---||---   steem   basic attention token
<u100>   ethereum   zcoin   stellar   0x   aeternity   ---||---   lisk   stellar
<u101>   nano   cybermiles   tether   bancor   aeternity   ---||---   ethereum   holo
<u103>   ethereum   dogecoin   tether   bancor   aeternity   ---||---   icon   power ledger
<u107>   electroneum   dogecoin   tether   bancor   aeternity   ---||---   icon   chainlink
<u119>   electroneum   dogecoin   tether   bancor   aeternity   ---||---   eos   horizen
<u120>   tron   holo   tether   bancor   aeternity   ---||---   eos   waves
<u125>   chainlink   holo   tether   bancor   aeternity   ---||---   ethereum   ripple
<u128>   crypto.com   neo   verge   tenx   funfair   ---||---   tenx   tezos
<u132>   ethereum   neo   stellar   litecoin   bitcoin cash   ---||---   ethereum   holo
<u147>   cybermiles   dentacoin   cardano   zcoin   tron   ---||---   ethereum   holo
<u148>   cybermiles   dentacoin   cardano   zcoin   tron   ---||---   ethereum   holo
<u152>   tether   cybermiles   bancor   aeternity   ark   ---||---   wanchain   tenx
<u154>   tether   cybermiles   bancor   aeternity   ark   ---||---   tenx   ripple
<u159>   stellar   cybermiles   bancor   aeternity   ark   ---||---   chainlink   komodo
<u166>   bytecoin   ethereum   litecoin   dash   eos   ---||---   bitcoin   ethereum
<u172>   dash   eos   zcoin   horizen   pivx   ---||---   basic attention token   steem
<u186>   holo   eos   zcoin   horizen   pivx   ---||---   horizen   waves
<u192>   kucoin shares   status   monacoin   funfair   eos   ---||---   holo   funfair
<u201>   ethereum   ripple   dash   bitcoin cash   lisk   ---||---   ethereum   verge
<u205>   bitcoin cash   tenx   ripple   holo   eos   ---||---   ethereum   verge
<u209>   bitcoin   ethereum   zcoin   stellar   0x   ---||---   verge   ethereum
<u211>   bitcoin   ethereum   zcoin   stellar   0x   ---||---   tenx   eos
<u214>   aion   zcoin   gas   verge   gxchain   ---||---   ethereum   ripple
<u218>   ethereum   dropil   gas   verge   gxchain   ---||---   ethereum   verge
<u219>   bitcoin   chainlink   gas   verge   gxchain   ---||---   tezos   ethereum
<u221>   bitcoin   ethereum   cardano   neo   verge   ---||---   bitcoin   tenx
<u222>   neo   stellar   litecoin   ripple   bitcoin cash   ---||---   horizen   eos
<u224>   iota   eos   komodo   maidsafecoin   metaverse etp   ---||---   tezos   maidsafecoin
<u226>   iota   eos   komodo   maidsafecoin   metaverse etp   ---||---   chainlink   horizen
```

## > Αξιολόγηση (Validation)

Αφού παραχθεί το εκτελέσιμο “ **recommendation** ”, με το όρισμα “ **-validate** ” εκτελείται η αξιολόγηση των αλγορίθμων, όπου το σφάλμα κυμαίνεται σε όλες τις εκτελέσεις από ~0.3 μέχρι ~0.4 (πολύ καλό εύρος σφάλματος ).
Παραθέτω κάποιες ενδεικτικές τιμές παρακάτω.

### Cosine LSH Recommendation MAE

```
Cosine LSH Recommendation MAE:

A. 0.3086247959     B. 0.3138353527
```

### Clustering Recommendation MAE

```
Clustering Recommendation MAE:

A. 0.3086532765     B. 0.3119283526
```