Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

«Ευφυές σύστημα πρόβλεψης τιμών μετοχών»

Παύλος Μαθιουδάκης
Γιοβάννα Τόσιτς

Επιβλέπων Καθηγητής: **Δρ. Νικόλαος Παπαδάκης**

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε όλους όσους στάθηκαν στο πλευρό μας όλο αυτό το χρονικό διάστημα. Πρώτα απ' όλα τις οικογένειές μας που μας στάθηκαν οικονομικά και ψυχολογικά παρέχοντάς μας την κατάλληλη στήριξη στις σπουδές μας. Θα θέλαμε να ευχαριστήσουμε επίσης όλο το κοινωνικό μας περίγυρο φίλους, συνάδελφους, διδάσκοντες που με τις γνώσεις τους και τη βοήθεια τους μας παρείχαν τα απαιτούμενα εφόδια για το μέλλον. Τέλος θα θέλαμε να αποδώσουμε τις ευχαριστίες μας στον εισηγητή και τον υπεύθυνο για την πτυχιακή μας εργασία κ. Νίκο Παπαδάκη για την ευκαιρία που μας έδωσε να ασχοληθούμε με το συγκεκριμένο θέμα.

Abstract

The subject of the thesis was to build an intelligent share price forecasting system using the C programming language and the use of algorithms Apriori and K-means. The system will offer users the ability to directly informed about the performance of the share that interests them, by correlating a share group.

The main idea lies in the fact that if someone buys a share and the correlation index with another share is high then there are great chances to buy the other share. As a result we have an increase of the share price. Then we will introduce step by step the process of creating the application and how it can be used.

The development was done in C, a basic programming language combined with the use of (.txt) files for storing data. We will refer to machine learning, specifically to the Apriori and K-means algorithms in some detail and the way they are programmed and work. You will become familiar with the use of the programming language C. Reference will be made on the popular Dev-C ++ tool as with it we programmed our system. We will make a detailed description of the use and options of the application and its capabilities as well.

Σύνοψη

Αντικείμενο της πτυχιακής εργασίας ήταν να κατασκευαστεί ένα ευφυΐες σύστημα πρόβλεψης τιμών μετοχών με την χρήση της γλώσσας προγραμματισμού C καθώς και την χρήση των αλγόριθμων Apriori και K-μέσων. Το σύστημα θα προσφέρει στους χρήστες τη δυνατότητα να ενημερώνονται άμεσα για την απόδοση της μετοχής που τους ενδιαφέρει, μέσω της συσχέτισης μιας ομάδας μετοχών. Η κύρια ιδέα βρίσκεται στο γεγονός ότι αν κάποιος αγοράζει κάποια μετοχή και ο δείκτης συσχέτισης της με κάποια άλλη μετοχή είναι υψηλός τότε συνεπάγεται ότι υπάρχουν μεγάλες πιθανότητες να αγοραστεί και η άλλη μετοχή. Σαν επακόλουθο έχουμε και την αύξηση της τιμής της μετοχής. Στη συνέχεια της πτυχιακής θα παρουσιάσουμε βήμα προς βήμα τη διαδικασία δημιουργίας της εφαρμογής και πως μπορεί αυτή να χρησιμοποιηθεί.
Η ανάπτυξη έγινε σε C, μια βασική γλώσσα προγραμματισμού σε συνδυασμό με την χρήση αρχείων(.txt) για την αποθήκευση των δεδομένων. Θα αναφερθούμε στην μηχανική μάθηση και ποιο συγκεκριμένα στους αλγορίθμους Apriori και K-μέσων με αρκετές λεπτομέρειες καθώς και τον τρόπο με τον οποίο προγραμματίζονται και δουλεύουν. Θα εξοικειωθούμε με την χρήση της γλώσσας προγραμματισμού C. Θα γίνει αναφορά στο δημοφιλές εργαλείο Dev-C++ καθώς με αυτό δημιουργήσαμε το σύστημα μας. Θα κάνουμε αναλυτική περιγραφή της χρήσης και των επιλογών της εφαρμογής αλλά και των δυνατοτήτων της.

Περιεχόμενα

Λίστα Πινάκων

Εισαγωγή στην C

Ιστορία της C

Η C είναι μια διαδικαστική γλώσσα προγραμματισμού γενικής χρήσης, η οποία αναπτύχθηκε αρχικά, μεταξύ του 1969 και του 1973, από τον Ντένις Ρίτσι στα εργαστήρια AT&T Bell Labs για να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX. Όπως οι περισσότερες διαδικαστικές γλώσσες προγραμματισμού που ακολουθούν την παράδοση της ALGOL, η C έχει δυνατότητες δομημένου προγραμματισμού και επιτρέπει τη χρήση αναδρομής (αλλά όχι και εμφωλευμένων συναρτήσεων), ενώ, ο στατικός ορισμός του τύπου των μεταβλητών που επιβάλλει, προλαμβάνει πολλά σφάλματα κατά την χρήση τους. Ο σχεδιασμός της περιλαμβάνει δομές που μεταφράζονται αποδοτικά σε τυπικές εντολές μηχανής (machine instructions) και εξ αιτίας αυτού χρησιμοποιείται συχνά σε εφαρμογές που παλιότερα γράφονταν σε συμβολική γλώσσα (assembly language). Αυτό ακριβώς το χαρακτηριστικό της, που έχει σαν συνέπεια και την αυξημένη ταχύτητα εκτέλεσης των εφαρμογών που γράφονται σε αυτή, καθώς και το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα, συνέβαλε κατά πολύ στην καθιέρωση της και την χρήση της για ανάπτυξη λειτουργικών συστημάτων και λοιπών προγραμμάτων συστήματος (system software), αλλά και απλών εφαρμογών.

Η C συγκαταλέγεται πλέον στις πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού όλων των εποχών και πολλές νεότερες γλώσσες έχουν επηρεαστεί άμεσα ή έμμεσα από αυτήν, συμπεριλαμβανομένων των C++, C#, D, Go,Java, JavaScript, Limbo, LPC, Perl, PHP, Python, καθώς και του κελύφους C (C shell) του Unix. Κάποιες από αυτές τις γλώσσες έχουν επηρεαστεί κυρίως στη σύνταξη τους, με το σύστημα τύπων, τα μοντέλα δεδομένων και το νόημα των εκφράσεων τους να διαφέρουν σημαντικά από την C. Η C++, ειδικά, ξεκίνησε σαν προεπεξεργαστής της C, αλλά έχει εξελιχθεί πλέον σε μια αντικειμενοστραφή γλώσσα, που αποτελεί υπερσύνολο της C.

Φιλοσοφία της γλώσσας C

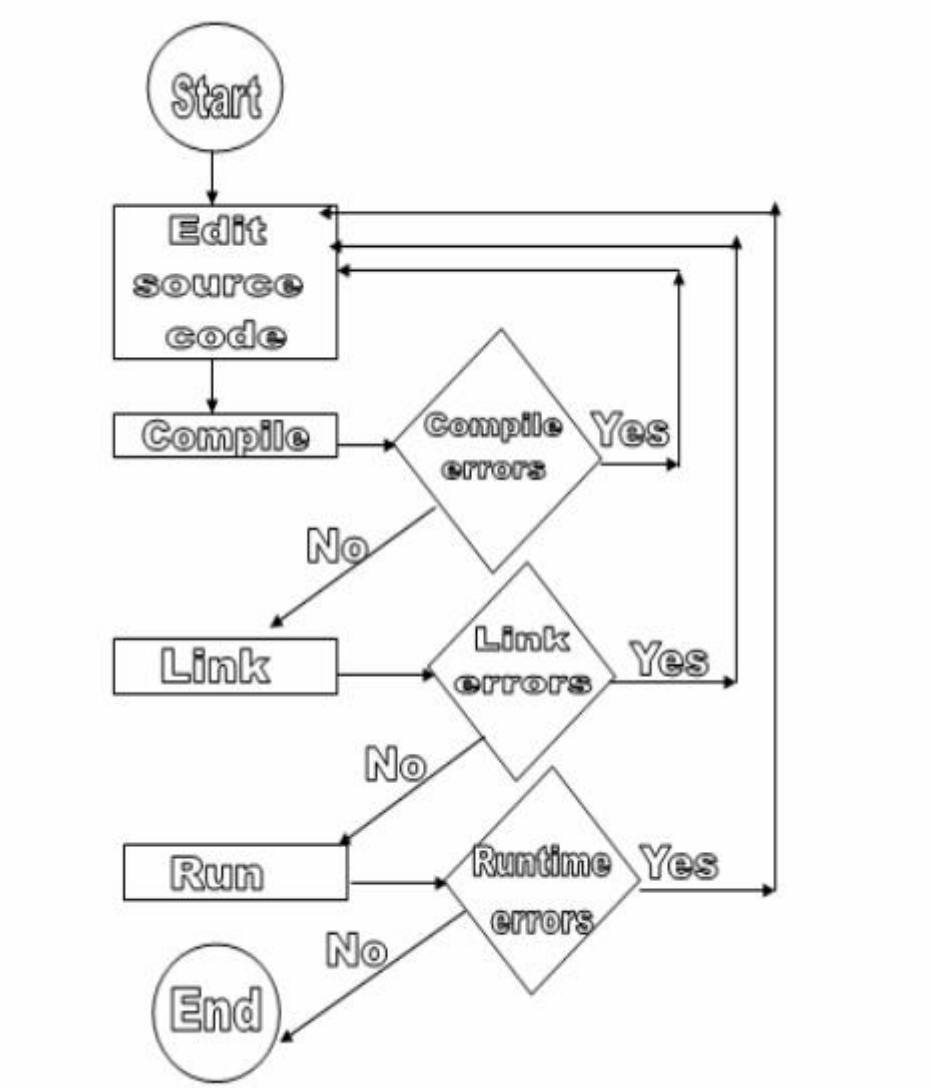
Η C είναι μια σχετικά μινιμαλιστική γλώσσα προγραμματισμού. Ανάμεσα στους σχεδιαστικούς στόχους που έπρεπε να καλύψει η γλώσσα περιλαμβανόταν το ότι θα μπορούσε να μεταγλωττιστεί άμεσα με τη χρήση μεταγλωττιστή ενός περάσματος (single-pass compiler) — με άλλα λόγια, ότι θα απαιτούνταν μόνο ένας μικρός αριθμός από εντολές σεγλώσσα μηχανής για κάθε βασικό στοιχείο της, χωρίς εκτεταμένη υποστήριξη στον χρόνο εκτέλεσης. Ως αποτέλεσμα, είναι δυνατό να γραφτεί κώδικας σε C σε χαμηλό επίπεδο προγραμματισμού με ακρίβεια ανάλογη της συμβολικής γλώσσας, στην πραγματικότητα η C ορισμένες φορές αποκαλείται (και χωρίς να υπάρχει πάντα αντιπαράθεση) «συμβολική γλώσσα υψηλού επιπέδου» («high-level assembly») ή «φορητή συμβολική γλώσσα» («portable assembly»). Επίσης, γίνονται αναφορές στη C ως γλώσσα προγραμματισμού μεσαίου επιπέδου.

Η C είναι μια ανώτερη γλώσσα προγραμματισμού. Δηλαδή μπορεί να την χρησιμοποιήσει κάποιος για να δημιουργήσει μια λίστα από εντολές (το πρόγραμμα ή τον κώδικα) που ο υπολογιστής θα ακολουθήσει σειριακά (υπάρχουν και παράλληλες εκδόσεις της γλώσσας σε platforms με περισσότερες από μια CPU's) για την λύση κάποιου προβλήματος. Η C δεν σημαίνει ότι είναι η καλύτερη υπάρχουσα γλώσσα προγραμματισμού, σίγουρα όμως είναι πλήρης και ευρύτατα διαδεδομένη. Για περισσότερα από 30 χρόνια εξελίσσεται και έχει γίνει πλήρως αποδεκτή από την κοινότητα των προγραμματιστών. Χρησιμοποιείται παντού από την δημιουργία προγραμμάτων για τον έλεγχο οικιακών συσκευών (micro-controllers) έως και την δημιουργία λειτουργικών συστημάτων (operating systems) όπως τα Windows XP το Linux ή το Unix.

• Με την C μπορεί να αποκτήσει κάποιος πλήρη έλεγχο πάνω σε υπολογιστικά συστήματα.

• Είναι αποτελεσματική στην λύση αριθμητικών ή όχι προβλημάτων.

• Τρέχει κάτω από όλα σχεδόν τα λειτουργικά συστήματα.



***Εικόνα 1**: Κύκλος Ανάπτυξης*

Η κατηγορία run time errors (σφάλματα κατά τον χρόνο εκτέλεσης) μπορεί να αναλυθεί σε 3 ειδών σφάλματα:

1. **Αριθμητικά σφάλματα**, για παράδειγμα διαίρεση με το 0.

2. **Σφάλματα υπερχείλισης (overflow)** που δημιουργούνται με τη λάθος χρησιμοποίηση του εύρους αναπαράστασης των μεταβλητών.

3. **Εννοιολογικά σφάλματα**. Ναι μεν το πρόγραμμα τρέχει χωρίς τεχνικά λάθη αλλά ο αλγόριθμος (ο τρόπος δηλαδή που επιλύουμε το εν λόγω πρόβλημα) είναι λανθασμένος.

Χαρακτηριστικά της C

Στη C δεν επιβάλλεται κάποια συγκεκριμένη μορφή στον πηγαίο κώδικα (όπως, για παράδειγμα, συνέβαινε στις αρχικές εκδόσεις της Fortran). Ο προγραμματιστής, χωρίς να αγνοεί φυσικά το συντακτικό της γλώσσας, είναι ελεύθερος να δώσει όποια μορφή θέλει στον κώδικα που γράφει (free-format source). Το ελληνικό ερωτηματικό (;) χρησιμοποιείται ως τερματιστής εντολών (και όχι ως διαχωριστής, όπως στην Pascal, παραδείγματος χάριν) και τα άγκιστρα ({}) χρησιμοποιούνται για την ομαδοποίηση εντολών (όπως τα begin/end στην Pascal).

Ακόμα, στη C όλος ο εκτελέσιμος κώδικας περιέχεται σε υπορουτίνες οι οποίες ονομάζονται «συναρτήσεις» (όχι με την αυστηρή έννοια του συναρτησιακού προγραμματισμού). Οι παράμετροι περνιούνται στις συναρτήσεις πάντα με τιμή (pass-by-value). Το πέρασμα με αναφορά (pass-by-reference) γίνεται έμμεσα στην ουσία, περνώντας, ως παραμέτρους των συναρτήσεων, δείκτες στις μεταβλητές των οποίων θέλουμε να αλλάζουμε τις τιμές μέσα από τις συναρτήσεις.

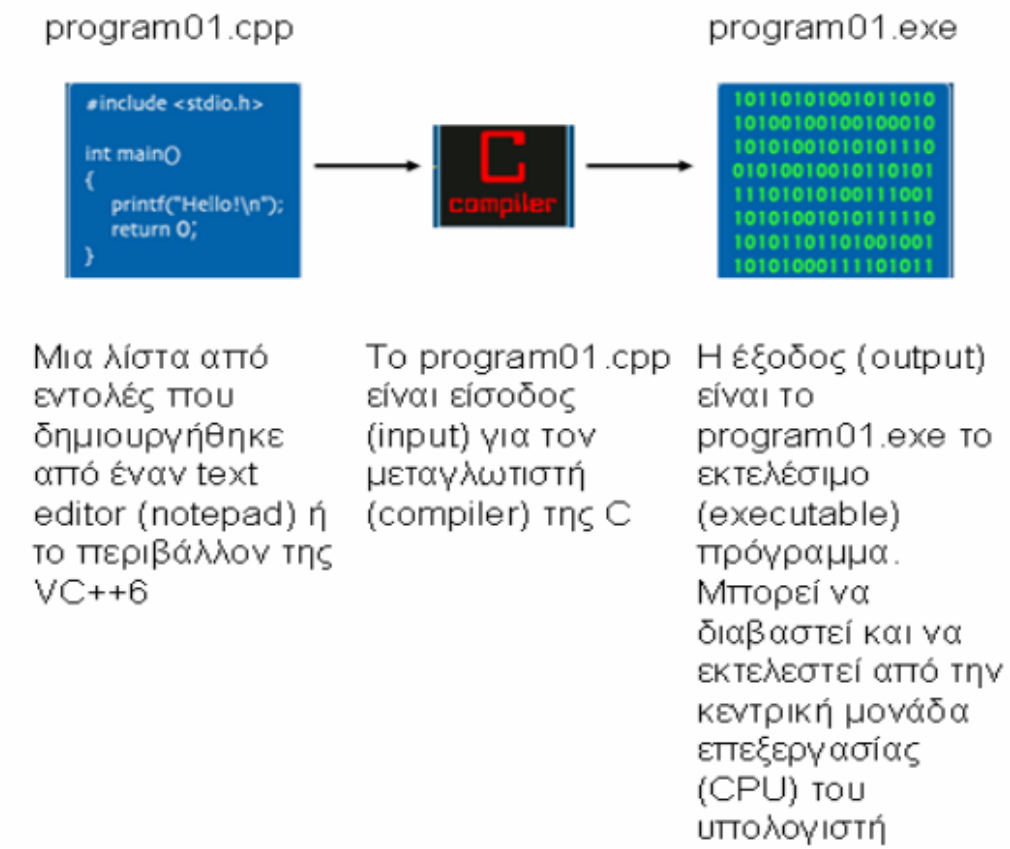
Η C έχει ακόμα τα εξής χαρακτηριστικά:

- Έχει ένα πολύ μικρό σταθερό πλήθος λέξεων-κλειδιών (keywords), το οποίο περιλαμβάνει ένα πλήρες σύνολο δομών/εντολών ελέγχου ροής: for, if/else, while, switch, και do/while, goto.
- Υπάρχει μόνο ένας χώρος ονομάτων (namespace) και τα ονόματα (μεταβλητών, συναρτήσεων, κ.τ.λ.) που ορίζονται από το χρήστη δεν διακρίνονται με κάποιο τρόπο από τις λέξεις-κλειδιά της γλώσσας.
- Υπάρχει ένα μεγάλο πλήθος αριθμητικών και λογικών τελεστών, όπως οι: +, +=, ++, -, -=, --, *, *=, /, /=, ==, &, &&, |, ||, ~, κ.ά.
- Σε μία εντολή μπορεί να γίνουν παραπάνω από μια εκχωρήσεις τιμών.
- Η τιμή που επιστρέφει μια συνάρτηση, μπορεί να αγνοηθεί εάν δεν χρειάζεται.
- Ο ορισμός των τύπων των μεταβλητών είναι στατικός και απαραίτητος, αλλά γίνονται έμμεσες μετατροπές από τη γλώσσα. Για παράδειγμα, παραστάσεις με τύπο χαρακτήρα μπορούν να χρησιμοποιηθούν σε σημεία που απαιτείται ακέραιος.
- Η σύνταξη των δηλώσεων ονομάτων προσομοιάζει την χρήση αυτών μέσα στον εκτελέσιμο κώδικα. Η C δεν έχει ειδική λέξη-κλειδί για τον ορισμό ονομάτων (όπως είναι η "var" στην Pascal, για παράδειγμα) ή συναρτήσεων (όπως η "function", πάλι στην Pascal). Μια γραμμή που ξεκινάει με το όνομα ενός τύπου, εκλαμβάνεται σαν ορισμός μεταβλητής ή συνάρτησης, ανάλογα με τον αν υπάρχουν, ή όχι, παρενθέσεις που περικλείουν (τυπικές) παραμέτρους συνάρτησης.
- Ο χρήστης μπορεί να ορίσει δικούς του τύπους (και σύνθετους), εάν το επιθυμεί. Μπορεί επίσης να ορίσει και τύπους εγγραφών (structs στη C, records σε άλλες γλώσσες).
- Μπορούν να οριστούν πίνακες, αν και δεν υπάρχει ειδική λέξη-κλειδί για τον ορισμό τους (όπως το "array" στην Pascal). Η δεικτοδότηση τους γίνεται με

χρήση αγκυλών ([]), αν και πολύ συχνά γίνεται χρήση αριθμητικής δεικτών. Το πρώτο στοιχείο κάθε πίνακα δεικτοδοτείται πάντα από το μηδέν (0). Π.χ.: Το στοιχείο month[0] είναι το πρώτο στοιχείο του πίνακα month. Τέλος, δεν υπάρχουν τελεστές για την σύγκριση ή εκχώρηση πινάκων.

- Είναι δυνατή η δημιουργία αριθμήσιμων τύπων με τη χρήση της λέξης-κλειδί "enum", οι οποίοι μπορούν να χρησιμοποιηθούν όπου οι ακέραιοι και αντίστροφα.
- Δεν υπάρχει ιδιαίτερος τύπος για αλφαριθμητικά, τα οποία παραδοσιακά υλοποιούνται και αντιμετωπίζονται σαν πίνακες από χαρακτήρες, και έχουν έναν μηδενικό χαρακτήρα να σημαδεύει το τέλος τους (null-terminated arrays of characters).
- Είναι δυνατή η άμεση προσπέλαση χαμηλού επιπέδου στη μνήμη του υπολογιστή με τη χρήση δεικτών.
- Οι υπορουτίνες που δεν επιστρέφουν τιμή ("procedures" σε άλλες γλώσσες) είναι συναρτήσεις που ορίζονται να είναι τύπου "void" (ψευδοτύπος που δείχνει την απουσία επιστρεφόμενης τιμής, αλλά χρησιμοποιείται και για δείκτες που δεν δείχνουν σε αντικείμενο συγκεκριμένου τύπου).
- Δεν μπορούν να οριστούν συναρτήσεις μέσα σε άλλες συναρτήσεις (εμφωλιασμένες).
- Οι δείκτες σε συναρτήσεις και δεδομένα επιτρέπουν την υλοποίηση πολυμορφισμού στην πράξη.
- Ο προ επεξεργαστής της γλώσσας επιτρέπει τον ορισμό μακροεντολών, την συγχώνευση αρχείων πηγαίου κώδικα, καθώς και την μεταγλώττιση υπό συνθήκες.
- Αρχεία πηγαίου κώδικα μπορούν να μεταγλωττιστούν χωριστά και να συνδεθούν μαζί, ενώ υπάρχει η δυνατότητα ελέγχου της ορατότητας συναρτήσεων και μεταβλητών στα άλλα αρχεία (πέρα από αυτό στο οποίο ορίζονται) με το χαρακτηρισμό τους ως "static" ή "extern".
- Οι πολύπλοκες λειτουργίες, όπως οι λειτουργίες εισόδου/εξόδου, ο χειρισμός των αλφαριθμητικών, καθώς και οι μαθηματικές συναρτήσεις, έχουν ανατεθεί, με συνεπή τρόπο, στις αντίστοιχες βιβλιοθήκες.

Η C δεν διαθέτει κάποιες από τις δυνατότητες νεότερων γλωσσών, όπως τον προσανατολισμό στα αντικείμενα και την συλλογή απορριμμάτων (garbage collection).



Εικόνα 1: *Δημιουργία προγράμματος σε C*

Τύποι δεδομένων

Η C έχει πέντε βασικούς τύπους δεδομένων:

char (character),

int (integer),

float (floating point),

double (double floating point),

void (no value).

Όλοι οι άλλοι τύποι της C βασίζονται σ' αυτούς. Όλοι οι βασικοί τύποι εκτός από τον τύπο void μπορεί ν' αλλάξουν γράφοντας πριν από τον τύπο τον κατάλληλο μετασχηματισμό. Οι μετασχηματισμοί αυτοί είναι οι**: signed, unsigned, long, και short.** Το μέγεθος και τα διαστήματα τιμών των τύπων της C εξαρτάται από τον επεξεργαστή. Στον πίνακα δίνουμε τους τύπους δεδομένων όπως ορίζονται από το πρότυπο ANSI.

Πίνακας 1*:Τύποι Δεδομένων*

Τύπος	μέγεθος σε bits	διάστημα τιμών
Char	8	-128..127
Unsigned char	8	0..255
Signed char	8	-127..127
Int	16	-32767..32767
Unsigned int	16	0..65535
Signed int	16	-32767..32767
Short int	16	-32767..32767
Unsigned short int	8	0..65535
Signed short int	8	-32767..32767
Long int	32	-2147483647..2147483647
Signed long int	32	0..4294967295
Unsigned long int	32	
Float	32	
Double	64	
Long double	128	

Δηλώσεις μεταβλητών

Τα αναγνωριστικά στη C μπορούν να έχουν όσους χαρακτήρες θέλουμε. Αν το αναγνωριστικό είναι εξωτερικό όνομα (όνομα συνάρτησης η καθολική μεταβλητή) τότε μόνο οι έξι πρώτοι χαρακτήρες είναι σημαντικοί διαφορετικά για εσωτερικά ονόματα οι πρώτοι 31 χαρακτήρες είναι σημαντικοί. **Τα κεφαλαία γράμματα στην C είναι διαφορετικά από τα μικρά.**

Η δήλωση μιας μεταβλητής έχει την γενική μορφή:

<τύπος> <λίστα μεταβλητών>

```
int i=0,j;
float f,g;
```

Δηλώσεις μεταβλητών κάνουμε μέσα σε συναρτήσεις (local variables, automatic), στις παραμέτρους μιας συνάρτησης (formal parameters) και έξω απ' όλες τις συναρτήσεις (global variables). Μπορούμε επίσης να δηλώσουμε μία τοπική μεταβλητή μέσα σε μία ενότητα π.χ.

```
 if (συνθήκη)
{
char x[30];
...
}
```

Στη περίπτωση αυτή η εμβέλεια της μεταβλητής είναι η ενότητα στην οποία είναι δηλωμένη. Έτσι αποφεύγουμε πλάγια αποτελέσματα και έχουμε οικονομία χώρου.

Καθολικές μεταβλητές έχουν εμβέλεια σε ολόκληρο το πρόγραμμα και δηλώνονται στην αρχή του κώδικα έξω απ' όλες τις συναρτήσεις. Όταν μια καθολική και μία τοπική μεταβλητή έχουν το ίδιο όνομα τότε μέσα στην εμβέλεια της τοπικής μεταβλητής αναφερόμαστε πάντα στην τοπική. Μεταβλητές που είναι παράμετροι συναρτήσεων συμπεριφέρονται ως τοπικές μεταβλητές της συνάρτησης.

Σταθερές εισάγονται με την προκαθορισμένη λέξη **const** π.χ.

```
const int i=1;
```

Πτητικές (volatile) μεταβλητές πληροφορούν τον μεταγλωττιστή ότι η τιμή τους μπορεί ν' αλλάξει χωρίς αυτό να δηλώνεται σαφώς στο πρόγραμμα.

Δηλώσεις σταθερών

Σταθερές χαρακτήρες γράφονται σε απλά εισαγωγικά.

char question_mark = '?';

Σταθερές τύπου string γράφονται μέσα σε διπλά εισαγωγικά, "---". Έτσι συνήθως τυπώνουμε διάφορα μηνύματα. Παραδείγματα σταθερών για τους άλλους τύπους δίνονται στον παρακάτω πίνακα.

int 1234
float 12.345F
double -0.9876544
char '?'

Υπάρχουν όμως ορισμένοι χαρακτήρες που δεν τυπώνονται με μια σταθερά τύπου string για παράδειγμα τα διπλά εισαγωγικά. Για τους χαρακτήρες αυτούς έχουμε τις λεγόμενες σταθερές backslash που δίνονται στον επόμενο πίνακα.

Πίνακας 2: *Σταθερές Backslash*

Κώδικας	Σημασία
\b \f \n \r \t \" \' \0 \\ \v \a \N \xN	backspace form feed νέα γραμμή carriage return οριζόντιο tab διπλά εισαγωγικά απλά εισαγωγικά Null backslash κάθετο tab alert οκταδική σταθερά δεκαεξαδική σταθερά

Τελεστές

Η C έχει τέσσερις τύπους τελεστών: **αριθμητικοί, σύγκρισης (συσχεσιακοί), λογικοί και τελεστές χειρισμού bits (bitwise operators).**Παρακάτω δίνουμε ένα πίνακα με όλους τους τελεστές διατεταγμένους σύμφωνα με την προτεραιότητά τους.

***Πίνακας 3**:Τελεστές διατεταγμένοι ανά προτεραιότητα*

Προτεραιότητα	Τελεστές
Υψηλότερη	( ) [ ] -> ! ~ ++ -- -(type) * & sizeof * / % - + << >> < <= > >= == != & ^ | && || ?
Χαμηλότερη	= += -= *= /=

Αριθμητικοί τελεστές

***Πίνακας 4**:Αριθμητικοί τελεστές*

-	αφαίρεση, πρόσημο
+	πρόσθεση
*	πολλαπλασιασμός
/	διαίρεση
%	(mod)
--	ελάττωση μεταβλητής κατά 1
++	αύξηση μεταβλητής κατά 1

Οι τελεστές -- και ++ μπορεί να τοποθετηθούν μπροστά ή μετά ένα τελεσταίο. Η εντολή --x; ισοδυναμεί με την x:=x-1 αλλά η αφαίρεση εκτελείται πριν χρησιμοποιήσουμε την τιμή της x. Όμοια και η εντολή ++x;

Η εντολή x--; ισοδυναμεί με την x:=x-1 αλλά η αφαίρεση εκτελείται αφού χρησιμοποιήσουμε την τιμή της x.

x=3; x=3;
y=++x; y=x++;
αποτέλεσμα: y=4 (x=4) αποτέλεσμα: y=3 (x=4)

Συσχεσιακοί και λογικοί τελεστές

Πίνακας 5: Συσχεσιακοί και λογικοί τελεστές

>	
>=	
<	
<=	
==	ίσον
!=	διάφορο του ίσον
&&	AND
||	OR
!	NOT

Στη C true είναι μια τιμή διαφορετική του μηδενός και false είναι το μηδέν.

Τελεστές χειρισμού bits (bitwise operators)

Χειρισμός των bits σημαίνει την δυνατότητα επέμβασης στα bits ενός byte ή μιας λέξης που αντιστοιχούν στους τύπους char και int.

&	AND
|	OR
^	XOR
~	συμπλήρωμα ως προς 1
>>	shift right
<<	shift left

Ο τελεστής << έχει τον τύπο: variable << number και μετακινεί τα όλα τα bits της μεταβλητής προς τ' αριστερά number θέσεις. Οι κενές θέσεις που δημιουργούνται από τα δεξιά αντικαθίστανται με 0. π.χ.

x=5; x: 0 0 0 0 0 1 0 1

x=x << 2; 0 0 0 1 0 1 0 0
(αποτέλεσμα x=20)

Συντομεύσεις (Shorthands)

Η C σε ορισμένες περιπτώσεις εκχώρησης μας επιτρέπει να συντομεύσουμε τον κώδικα. Έτσι μία παράσταση της μορφής:

<μεταβλητή> =<μεταβλητή> <τελεστής><παράσταση>

μπορεί να γραφτεί ως:
<μεταβλητή> <τελεστής> = <παράσταση>

Για παράδειγμα οι εντολές:

year+=11; year=year+11;
i-=3; ισοδυναμούν με i=i-3;
x*=10; x=x*10;
y/=5; y=y/5;

Παράδειγμα “Hello World”

Πρόγραμμα σε C που εκτυπώνει στο τερματικό "Hello world":

```
#include <stdio.h>
int main( int argc, char **argv ) {
  printf( "Hello world!" );
  return 0;
}
```

Μηχανική μάθηση

Μηχανική μάθηση είναι υποπεδίο της επιστήμης των υπολογιστών που αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Το 1959, ο Arthur Samuel ορίζει τη μηχανική μάθηση ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί". Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Η μηχανική μάθηση είναι στενά συνδεδεμένη και συχνά συγχέεται με υπολογιστική στατιστική, ένας κλάδος, που επίσης επικεντρώνεται στην πρόβλεψη μέσω της χρήσης των υπολογιστών. Έχει ισχυρούς δεσμούς με την μαθηματική βελτιστοποίηση, η οποία παρέχει μεθόδους, τη θεωρία και τομείς εφαρμογής. Η Μηχανική μάθηση εφαρμόζεται σε μια σειρά από υπολογιστικές εργασίες, όπου τόσο ο σχεδιασμός όσο και ο ρητός προγραμματισμός των αλγορίθμων είναι ανέφικτος. Παραδείγματα εφαρμογών αποτελούν τα φίλτρα spam (spam filtering), η οπτική αναγνώριση χαρακτήρων (OCR), οι μηχανές αναζήτησης και η υπολογιστική όραση. Η Μηχανική μάθηση μερικές φορές συγχέεται με την εξόρυξη δεδομένων, όπου η τελευταία επικεντρώνεται περισσότερο στην εξερευνητική ανάλυση των δεδομένων, γνωστή και ως μη επιτηρούμενη μάθηση.

Στο πεδίο της ανάλυσης δεδομένων, η μηχανική μάθηση είναι μια μέθοδος που χρησιμοποιείται για την επινόηση πολύπλοκων μοντέλων και αλγορίθμων που οδηγούν στην πρόβλεψη. Τα αναλυτικά μοντέλα επιτρέπουν στους ερευνητές, τους επιστήμονες δεδομένων, τους μηχανικούς και τους αναλυτές να παράγουν αξιόπιστες αποφάσεις και αποτελέσματα και να αναδείξουν αλληλοσυσχετίσεις μέσω της μάθησης από ιστορικές σχέσεις και τάσεις στα δεδομένα.

Ορισμός

Ο Tom M. Mitchell πρότεινε έναν πιο επίσημο ορισμό που χρησιμοποιείται ευρέως: «Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία E ως προς μια κλάση εργασιών T και ένα μέτρο επίδοσης P, αν η επίδοσή του σε εργασίες της κλάσης T, όπως αποτιμάται από το μέτρο P, βελτιώνεται με την εμπειρία E». Αυτός ο ορισμός είναι σημαντικός για τον καθορισμό της μηχανικής μάθησης σε βασικό λειτουργικό πλαίσιο παρά με γνωστικούς όρους, ακολουθώντας έτσι την πρόταση του Alan Turing στην εργασία του «Υπολογιστικές μηχανές και Νοημοσύνη», ότι το ερώτημα αν μπορούν οι μηχανές να σκεφτούν, μπορεί να αντικατασταθεί με το ερώτημα αν μπορούν οι μηχανές να κάνουν αυτό που εμείς (ως σκεπτόμενες οντότητες) μπορούμε να κάνουμε.

Τύποι προβλημάτων και εργασιών

Οι εργασίες μηχανικής μάθησης συνήθως ταξινομούνται σε τρεις μεγάλες κατηγορίες, ανάλογα με τη φύση του εκπαιδευτικού «σήματος» ή την «ανατροφοδότηση» που είναι διαθέσιμα σε ένα σύστημα εκμάθησης.

Αυτές είναι:

- Επιτηρούμενη μάθηση (αλλιώς επιβλεπόμενη μάθηση ή μάθηση με επίβλεψη) (supervised learning): Το υπολογιστικό πρόγραμμα δέχεται τις παραδειγματικές εισόδους καθώς και τα επιθυμητά αποτελέσματα από έναν «δάσκαλο», και ο στόχος είναι να μάθει έναν γενικό κανόνα προκειμένου να αντιστοιχίσει τις εισόδους με τα αποτελέσματα.
- Μη επιτηρούμενη μάθηση (αλλιώς επίβλεπτη μάθηση ή μάθηση χωρίς επίβλεψη ()unsupervised learning): Χωρίς να παρέχεται κάποια εμπειρία στον αλγόριθμο μάθησης, πρέπει να βρει την δομή των δεδομένων εισόδου. Η Μη επιτηρούμενη μάθηση μπορεί να είναι αυτοσκοπός (ανακαλύπτοντας κρυμμένα μοτίβα σε δεδομένα) ή μέσο για ένα τέλος (χαρακτηριστικό της μάθησης).
- Ενισχυτική μάθηση: Ένα πρόγραμμα υπολογιστή αλληλοεπιδρά με ένα δυναμικό περιβάλλον στο οποίο πρέπει να επιτευχθεί ένας συγκεκριμένος στόχος (όπως η οδήγηση ενός οχήματος), χωρίς κάποιος δάσκαλος να του λέει ρητά αν έχει φτάσει κοντά στο στόχο του. Ένα άλλο παράδειγμα είναι να μάθει να παίζει ένα παιχνίδι εναντίον κάποιου αντιπάλου.

Μεταξύ της επιτηρούμενης και της μη επιτηρούμενης μάθησης είναι η ημι-επιτηρούμενη μάθηση, όπου ο δάσκαλος δίνει ένα ελλιπές εκπαιδευτικό σήμα: ένα σύνολο εκπαίδευσης με κάποια (συχνά πολλά) από τα αποτελέσματα στόχους λείπουν. Η Μεταγωγή είναι μια ειδική περίπτωση της αρχής αυτής, όπου το σύνολο των καταστάσεων του προβλήματος είναι γνωστό κατά το χρόνο εκμάθησης, όμως ένα μέρος των στόχων λείπουν.

Μεταξύ άλλων κατηγοριών μηχανικής μάθησης, υπάρχει ακόμα διαδικασία εκμάθησης (meta learning) που μαθαίνει στην μηχανή (να αναπτύσσει) τις δικές της επαγωγικές μεθόδους, βασιζόμενο στην προηγούμενη εμπειρία. Η Αναπτυξιακή μάθηση (Developmental robotics), η οποία έχει αναπτυχθεί για την εκμάθηση από ρομπότ, δημιουργεί τη δική της ακολουθία μαθησιακών καταστάσεων, ώστε το ρομπότ συσσωρευτικά αποκτά ποικιλία δεξιοτήτων μέσω της αυτόνομης αυτοεξερεύνησης και της κοινωνικής αλληλεπίδρασης με ανθρώπους εκπαιδευτές και χρησιμοποιώντας μηχανισμούς καθοδήγησης, όπως η ενεργητική μάθηση, η ωρίμανση και η μίμηση.

Μια άλλη κατηγοριοποίηση των προβλημάτων μηχανικής μάθησης προκύπτει όταν κάποιος θεωρήσει το επιθυμητό αποτέλεσμα του συστήματος μηχανικής μάθησης.

- Στην ταξινόμηση, τα δεδομένα εισόδου χωρίζονται σε δύο ή περισσότερες κλάσεις, και η μηχανή πρέπει να κατασκευάσει ένα μοντέλο, το οποίο θα αντιστοιχίζει τα δεδομένα σε μία ή περισσότερες (multi-label ταξινόμηση) κλάσεις. Αυτό συνήθως εμπίπτει στην επιτηρούμενη μάθηση. Τα φίλτρα Spam είναι ένα παράδειγμα ταξινόμησης, όπου οι είσοδοι είναι τα emails ή άλλα μηνύματα και οι κλάσεις είναι "spam" και "όχι spam".
- Στην παλινδρόμηση, επίσης πρόβλημα επιτηρούμενης μάθησης, τα αποτελέσματα είναι συνεχή και όχι διακριτά.

- Στην συσταδοποίηση, ένα σύνολο εισόδων πρόκειται να χωριστεί σε ομάδες. Σε αντίθεση με την ταξινόμηση, οι ομάδες δεν είναι γνωστές εκ των προτέρων, καθιστώντας αυτόν τον διαχωρισμό τυπική εργασία μη επιτηρούμενης μάθησης.
- Στην εκτίμηση πυκνότητας βρίσκει την κατανομή των δεδομένων εισόδου σε κάποιο χώρο.
- Σε προβλήματα μείωσης διαστασιμότητας (dimensionality reduction), τα δεδομένα απλοποιούνται και αντιστοιχίζονται σε ένα χώρο λιγότερων διαστάσεων. Το στατιστικό μοντέλο θεμάτων (Topic modeling) είναι ένα σχετικό πρόβλημα, όπου η μηχανή καλείται να βρει έγγραφα που καλύπτουν παρόμοια θέματα από ένα σύνολο εγγράφων γραμμένων σε φυσική γλώσσα.

Ιστορία και σχέσεις με άλλους τομείς

Ως επιστημονικό εγχείρημα, η μηχανική μάθηση αναπτύχθηκε από την αναζήτηση για την τεχνητή νοημοσύνη. Ήδη από την πρώιμη περίοδο της έρευνας στον τομέα της τεχνητής νοημοσύνης σε ακαδημαϊκό επίπεδο, το ζήτημα της κατασκευής μηχανών που θα μάθαιναν από δεδομένα απασχόλησε τους ερευνητές. Προσπάθησαν να προσεγγίσουν το πρόβλημα με διάφορες συμβολικές μεθόδους, καθώς και με τα λεγόμενα νευρωνικά δίκτυα. Αυτά ήταν ως επί το πλείστον perceptrons και μοντέλα, που όπως διαπιστώθηκε αργότερα ήταν επανεφευρέσεις των γενικευμένων γραμμικών μοντέλων της στατιστικής. Επίσης χρησιμοποιήθηκε η πιθανοθεωρητική λογική, ιδιαίτερα στην αυτοματοποιημένη ιατρική διάγνωση.

Ωστόσο, μια αυξανόμενη έμφαση σε προσεγγίσεις που βασίζονται στην λογική γνώση προκάλεσε ένα ρήγμα μεταξύ Τεχνητής Νοημοσύνης και μηχανικής μάθησης. Τα πιθανοθεωρητικά συστήματα μαστίζονταν από θεωρητικά και πρακτικά προβλήματα απόκτησης δεδομένων και αναπαράστασής τους. Από το 1980, έμπειρα συστήματα επικράτησαν στο πεδίο της Τεχνητής Νοημοσύνης, και ο ρόλος της στατιστικής υποχώρησε. Η εργασία σε συμβολική/βασισμένη σε γνώση εκμάθηση συνεχίστηκε εντός της ΤΝ, οδηγώντας στον επαγωγικό λογικό προγραμματισμό, αλλά οι κατευθυντήριες γραμμές της στατιστικής ήταν τώρα έξω από το χώρο της τεχνητής νοημοσύνης, στην αναγνώριση προτύπων και στην ανάκτηση πληροφοριών. Η έρευνα για νευρωνικά δίκτυα εγκαταλείφτηκε από την ΤΝ και την Επιστήμη Υπολογιστών τον ίδιο περίπου καιρό. Η ίδια επίσης κατεύθυνση ακολουθήθηκε πέρα από την ΤΝ και την πληροφορική, από ερευνητές άλλων ειδικοτήτων, συμπεριλαμβανομένων των Hopfield, Rumelhart και Hinton. Η επιτυχία ήρθε στα μέσα της δεκαετίας του 1980 με την επανεφεύρεση της μεθόδου ανάστροφης μετάδοσης (backpropagation).

Η Μηχανική μάθηση, αναδιοργανώθηκε ως ένα ξεχωριστό πεδίο, που άρχισε να ακμάζει κατά τη δεκαετία του 1990. Η προσοχή μετατοπίστηκε από τις συμβολικές προσεγγίσεις που κληρονόμησε από την Τεχνητή Νοημοσύνη, που στόχο είχαν την αντιμετώπιση επιλύσιμων προβλημάτων πρακτικής φύσης, και δόθηκε έμφαση σε μεθόδους και μοντέλα της στατιστικής και της θεωρίας πιθανοτήτων. Επίσης επωφελήθηκε από την διαθεσιμότητα ψηφιοποιημένων πληροφοριών και της δυνατότητας να διανεμηθούν μέσω του Διαδικτύου.

Η Μηχανική μάθηση και η εξόρυξη δεδομένων συχνά χρησιμοποιούν τις ίδιες μεθόδους και επικαλύπτονται σημαντικά. Μπορούν να διακριθούν ως εξής:

- Η μηχανική μάθηση εστιάζει στην πρόβλεψη, που βασίζεται σε γνωστές ιδιότητες που απορρέουν από το σύνολο εκπαίδευσης.
- Η εξόρυξη δεδομένων εστιάζει στην ανακάλυψη ιδιοτήτων μη γνωστών εκ των προτέρων. Αυτό είναι το βήμα ανάλυσης στην Ανακάλυψη Γνώσης από βάσεις δεδομένων

Οι δύο τομείς επικαλύπτονται με πολλούς τρόπους. Η εξόρυξη δεδομένων χρησιμοποιεί πολλές μεθόδους μηχανικής μάθησης, αλλά συχνά με διαφορετικούς στόχους. Από την άλλη πλευρά και η μηχανική μάθηση χρησιμοποιεί μεθόδους εξόρυξης δεδομένων, όπως η μη επιτηρούμενη μάθηση, ή στο στάδιο προεπεξεργασίας για να βελτιώνει την ακρίβεια της μάθησης. Ένα μεγάλο μέρος της σύγχυσης μεταξύ των δύο ερευνητικών τομέων (που συχνά έχουν ξεχωριστά συνέδρια και περιοδικά, με το ECML PKDD να αποτελεί σημαντική εξαίρεση) προκύπτει από τις βασικές υποθέσεις πάνω στις οποίες και οι δύο δουλεύουν. Όμως, στην μηχανική μάθηση η απόδοση συνήθως αξιολογείται ως προς την ικανότητα αναπαραγωγής γνώσης, την οποία ήδη κατέχουμε, ενώ στην ανακάλυψη γνώσης και την εξόρυξη δεδομένων το κλειδί είναι η ανακάλυψη γνώσης που δεν προκατέχουμε. Στην πρώτη περίπτωση μια μέθοδος επιτηρούμενης μάθησης μπορεί να έχει καλύτερα αποτελέσματα, ενώ σε μία τυπική διεργασία Ανακάλυψης Γνώσης και Εξόρυξης δεδομένων οι επιτηρούμενες μέθοδοι μάθησης δεν λειτουργούν εξαιτίας της μη διαθεσιμότητας συνόλου εκπαίδευσης.

Η μηχανική μάθηση συνδέεται επίσης με την βελτιστοποίηση: πολλά προβλήματα μάθησης διατυπώνονται ως η ελαχιστοποίηση της συνάρτησης απώλειας από ένα σύνολο δεδομένων εκπαίδευσης. Η συνάρτηση απώλειας εκφράζει τη διαφορά μεταξύ των προβλέψεων του εκπαιδευμένου μοντέλου και των πραγματικών καταστάσεων του προβλήματος. Η διαφορά των δύο τομέων απορρέει από τον στόχο της γενίκευσης: ενώ οι αλγόριθμοι βελτιστοποίησης μπορούν να ελαχιστοποιήσουν την απώλεια ενός συνόλου εκπαίδευσης, η μηχανική μάθηση εστιάζει στην ελαχιστοποίηση της απώλειας σε άγνωστες καταστάσεις.

Σχέση με την στατιστική

Η μηχανική μάθηση και η στατιστική είναι δύο στενά συνδεδεμένοι επιστημονικοί τομείς. Σύμφωνα με τον Michael Jordan, οι ιδέες της μηχανικής μάθησης, από τις μεθοδολογικές αρχές μέχρι τα θεωρητικά εργαλεία, προϋπάρχουν στην στατιστική. Ο ίδιος επίσης πρότεινε τον όρο Επιστήμη Δεδομένων για το συνολικό πεδίο

Ο Leo Breiman διέκρινε δύο υποδείγματα στατιστικής μοντελοποίησης: το μοντέλο δεδομένων και το αλγοριθμικό μοντέλο. Το αλγοριθμικό μοντέλο ταυτίζεται σχεδόν με αλγορίθμους μηχανικής μάθησης όπως τα Τυχαία Δάση.

Τέλος, ορισμένοι στατιστικολόγοι υιοθετούν μεθόδους μηχανικής μάθησης, με αποτέλεσμα την δημιουργία ενός ανασυνδεμένου τομέα που ονομάζεται στατιστική μάθηση.

Θεωρία της Μηχανικής Μάθησης

Ο βασικός στόχος ενός μαθητευόμενου είναι να γενικεύει την εμπειρία του. Σε αυτό το πλαίσιο γενίκευση είναι η ικανότητα μιας μηχανής μάθησης να αποδίδει με ακρίβεια σε καινούριες, πρωτόγνωρες εργασίες, αφού πρώτα έχει εκπαιδευτεί σε ένα σύνολο δεδομένων εκπαίδευσης. Γενικά τα προς εκπαίδευση παραδείγματα προέρχονται από κάποια άγνωστη κατανομή πιθανότητας, η οποία θεωρείται αντιπροσωπευτική του χώρου των καταστάσεων, και η μηχανή πρέπει να κατασκευάσει ένα γενικό μοντέλο που θα επιτρέπει την παραγωγή προβλέψεων σε καινούριες καταστάσεις με επαρκή ακρίβεια.

Η υπολογιστική ανάλυση των αλγορίθμων των μηχανών μάθησης και η απόδοσή τους είναι ένας κλάδος της θεωρητικής πληροφορικής, γνωστός ως Υπολογιστική θεωρία μάθησης. Επειδή τα εκπαιδευτικά σύνολα είναι πεπερασμένα και το μέλλον αβέβαιο, η θεωρία μάθησης δεν εγγυάται πάντα την απόδοση των αλγορίθμων. Αντ΄αυτού είναι συχνή η χρήση των πιθανοθεωρητικών ορίων της απόδοσης.

Το πόσο καλά ένα μοντέλο, που έχει εκπαιδευτεί σε υπαρκτά παραδείγματα, μπορεί να προβλέψει άγνωστες καταστάσεις ονομάζεται γενίκευση. Για την καλύτερη δυνατή γενίκευση, η πολυπλοκότητα της υπόθεσης θα πρέπει να είναι αντίστοιχη της πολυπλοκότητας της συνάρτησης των δεδομένων.

Πέρα όμως από την απόδοση, οι θεωρητικοί της υπολογιστικής μάθησης μελετούν την χρονική πολυπλοκότητα καθώς και το κατά πόσο είναι εφικτή η μάθηση. Στην υπολογιστική θεωρία μάθησης ένας υπολογισμός θεωρείται εφικτός αν μπορεί να επιτελεστεί σε πολυωνυμικό χρόνο. Υπάρχουν δύο είδη αποτελεσμάτων αναφορικά με την χρονική πολυπλοκότητα. Τα θετικά αποτελέσματα που σημαίνουν ότι μια συγκεκριμένη κλάση αντιστοιχίσεων μπορούν να επιτευχθούν σε πολυωνυμικό χρόνο και τα αρνητικά αποτελέσματα που δείχνουν το αντίθετο.

Υπάρχουν πολλές ομοιότητες μεταξύ της μηχανικής μάθησης και της στατιστικής συμπερασματολογίας, αν και χρησιμοποιούν διαφορετικούς όρους.

Προσεγγίσεις

Εκμάθηση με δέντρο απόφασης: Η εκμάθηση με δέντρο απόφασης χρησιμοποιεί ένα δέντρο απόφασης ως προγνωστικό μοντέλο, το οποίο αντιστοιχίζει παρατηρήσεις σχετικά με ένα στοιχείο σε συμπεράσματα σχετικά με την τιμή στόχο του αντικειμένου.

Εκμάθηση με Κανόνες συσχέτισης: Η εκμάθηση με κανόνες συσχέτισης είναι μια μέθοδος ανακάλυψης ενδιαφέρων σχέσεων μεταξύ των μεταβλητών σε μεγάλες βάσεις δεδομένων.

Τεχνητά νευρωνικά δίκτυα: Ένας αλγόριθμος εκμάθησης Τεχνητού νευρωνικού δικτύου, που συνήθως ονομάζεται "νευρωνικό δίκτυο" (NN), είναι ένας αλγόριθμος μάθησης, που εμπνέεται από τη δομή και τις λειτουργικές πτυχές των βιολογικών νευρωνικών δικτύων. Η δομή των υπολογισμών βασίζεται σε μια ομάδα εσωτερικά διασυνδεδεμένων τεχνητών νευρώνων, οι οποίοι επεξεργάζονται την πληροφορία και εκτελούν υπολογισμούς επικοινωνώντας μεταξύ τους. Τα σύγχρονα νευρωνικά δίκτυα είναι εργαλεία μη γραμμικής στατιστικής μοντελοποίησης δεδομένων. Συνήθως χρησιμοποιούνται για τη μοντελοποίηση σύνθετων σχέσεων μεταξύ δεδομένων εισόδου και εξόδου, για την ανακάλυψη προτύπων στα δεδομένα, ή για τον εντοπισμό στατιστικής δομής σε μία άγνωστη κοινή κατανομή πιθανότητας μεταξύ των παρατηρούμενων μεταβλητών.

Βαθιά Μάθηση: Η πτώση των τιμών του υλικού των τελευταίων ετών καθώς και η ανάπτυξη των GPU για προσωπική χρήση, οδήγησε στην ανάπτυξη της ιδέας της Βαθιάς Μάθησης. Αυτή η προσέγγιση προσπαθεί να μοντελοποιήσει τον τρόπο που ο ανθρώπινος εγκέφαλος επεξεργάζεται το φως και τον ήχο και τα μετατρέπει σε όραση και ακοή. Ορισμένες επιτυχείς εφαρμογές της Βαθιάς μάθησης είναι η μηχανική όραση και η αναγνώριση ομιλίας.

Επαγωγικός λογικός προγραμματισμός: Ο Επαγωγικός λογικός προγραμματισμός (ILP) είναι μια προσέγγιση που διέπει την μάθηση και χρησιμοποιεί λογικό προγραμματισμό ως τρόπο παρουσίασης των παραδειγμάτων εισόδου, του γνωστικού υποβάθρου και των υποθέσεων. Δεδομένης μιας κωδικοποίησης του γνωστικού υποβάθρου και ενός συνόλου παραδειγμάτων που παρουσιάζονται σαν λογική βάση γεγονότων, το σύστημα ΕΛΠ παράγει το υποτιθέμενο λογικό πρόγραμμα που περιέχει όλα τα θετικά και κανένα αρνητικό παράδειγμα. Ο επαγωγικός προγραμματισμός είναι ένας σχετικός τομέας που λαμβάνει υπόψιν κάθε είδος προγραμματιστικής γλώσσας για την αναπαράσταση υποθέσεων (και όχι μόνο λογικό προγραμματισμό), όπως τα συναρτησιακά προγράμματα.

Μηχανές διανυσμάτων υποστήριξης: Οι μηχανές διανυσμάτων υποστήριξης είναι ένα σύνολο μεθόδων επιτηρούμενης μάθησης που χρησιμοποιούνται για την ταξινόμηση και την παλινδρόμηση. Σ' αυτήν την περίπτωση δίνεται ένα σύνολο παραδειγμάτων εκπαίδευσης και κάθε φορά δηλώνεται σε ποια από τις δύο κατηγορίες ανήκει το παράδειγμα. Μία μηχανή διανυσμάτων υποστήριξης κατασκευάζει ένα μοντέλο που προβλέπει αν το νέο παράδειγμα εμπίπτει στην μία κατηγορία ή την άλλη.

Ομαδοποίηση: Η ομαδοποίηση είναι η διαδικασία κατά την οποία ένα σύνολο παρατηρήσεων χωρίζεται σε υποσύνολα έτσι ώστε οι παρατηρήσεις που ανήκουν στην ίδια ομάδα (cluster) είναι όμοιες, σύμφωνα με κάποιο ή κάποια προκαθορισμένα κριτήρια, ενώ οι παρατηρήσεις που προέρχονται από διαφορετικά υποσύνολα είναι ανόμοιες. Διαφορετικές τεχνικές κατηγοριοποίησης οδηγούν σε διαφορετικές υποθέσεις σχετικά με τη δομή των δεδομένων, οι οποίες συχνά καθορίζονται από κάποιο μέτρο ομοιότητας και αξιολογούνται για παράδειγμα ως προς την εσωτερική συνοχή (ομοιότητα μεταξύ των μελών του ίδιου cluster) και το διαχωρισμό ανάμεσα σε διαφορετικές ομάδες. Άλλες μέθοδοι βασίζονται στην εκτιμώμενη πυκνότητα και την συνεκτικότητα των γραφημάτων. Η ομαδοποίηση είναι μία μέθοδος μη επιτηρούμενης μάθησης και μία τεχνική η οποία χρησιμοποιείται επίσης στην στατιστική ανάλυση δεδομένων.

Δίκτυα Bayes: Ένα δίκτυο Bayes, ένα δίκτυο εμπιστοσύνης ή ένα άκυκλο γραφικό μοντέλο είναι ένα πιθανοθεωρητικό γραφικό μοντέλο που απεικονίζει ένα σύνολο τυχαίων μεταβλητών και την μεταξύ τους υποθετική ανεξαρτησία διαμέσου ενός κατευθυνόμενου άκυκλου γράφου. Για παράδειγμα, ένα δίκτυο Bayes μπορεί να αναπαραστήσει την πιθανοθεωρητική σχέση μεταξύ ασθενειών και συμπτωμάτων. Δεδομένων των συμπτωμάτων, το δίκτυο μπορεί να χρησιμοποιηθεί για να υπολογίσει τις πιθανότητες παρουσίας διαφόρων ασθενειών.

Ενισχυτική μάθηση: Η Ενισχυτική μάθηση ασχολείται με το πώς ένα υποκείμενο (πράκτορας) θα πρέπει να δράσει σε ένα περιβάλλον , έτσι ώστε να μεγιστοποιηθεί κάποια έννοια μακροπρόθεσμης ανταμοιβής. Οι αλγόριθμοι ενισχυτικής μάθησης προσπαθούν να βρουν μια πολιτική που αντιστοιχίζει τις καταστάσεις του περιβάλλοντος με τις ενέργειες που ο πράκτορας θα πρέπει να επιτελέσει σε αυτές τις καταστάσεις. Η ενισχυτική μάθηση διαφέρει από τα προβλήματα επιτηρούμενης μάθησης αφού τα σωστά ζεύγη δεδομένων εισόδου/εξόδου ζεύγη δεν παρουσιάστηκαν ποτέ, ούτε οι βέλτιστες δυνατές ενέργειες έχουν ρητά διορθωθεί.

Εκμάθηση με μέτρο ομοιότητας: Σε αυτή την κατηγορία προβλημάτων δίνονται στην μηχανή μάθησης ζεύγη παραδειγμάτων που θεωρούνται όμοια και ζεύγη που θεωρούνται ανόμοια. Τότε η μηχανή μάθησης πρέπει να μάθει μια συνάρτηση ομοιότητας (ή μια συνάρτηση μετρικής απόστασης), που μπορεί να προβλέψει αν δύο καινούρια αντικείμενα είναι όμοια. Πρόκειται για μια τεχνική που χρησιμοποιείται σε συστήματα σύστασης.

Γενετικοί αλγόριθμοι: Ένας γενετικός αλγόριθμος (GA) είναι μια ευρετική αναζήτηση που μιμείται τη διαδικασία της φυσικής επιλογής, και χρησιμοποιεί μεθόδους όπως αυτή της μετάλλαξης και της διασταύρωσης προκειμένου να δημιουργήσει καινούρια γονότυπα με την ελπίδα εύρεσης αποτελεσματικών λύσεων σε ένα συγκεκριμένο πρόβλημα. Στη μηχανική μάθηση, γενετικοί αλγόριθμοι χρησιμοποιήθηκαν τη δεκαετία του 1980 και του 1990. Αντίστροφα, τεχνικές μηχανικής μάθησης έχουν χρησιμοποιηθεί για την βελτίωση της απόδοσης γενετικών και εξελικτικών αλγορίθμων.

Εφαρμογές

Οι εφαρμογές της Μηχανικής Μάθησης περιλαμβάνουν:

- Αναγνώριση ομιλίας και γραφικού χαρακτήρα
- Ανάκτηση πληροφορίας
- Βελτιστοποίηση
- Βιοπληροφορική
- Διαδυκτιακή Διαφήμιση
- Εντοπισμός Διαδυκτιακής απάτης
- Εντοπισμός απάτης πιστωτικής κάρτας
- Επεξεργασία φυσικής γλώσσας
- Ηλεκτρονικά παιχνίδια
- Ιατρική Διάγνωση
- Κατηγοριοποίηση ακολουθιών DNA
- Λογισμικά
- Μάρκετινγκ
- Μετακίνηση Ρομπότ
- Μηχανές αναζήτησης
- Μηχανική αντίληψη
- Οικονομία
- Συναισθηματική υπολογιστική

- Συστήματα σύστασης
- Υπολογιστική ανατομία
- Υπολογιστική όραση - συμπεριλαμβανομένης της αναγνώρισης αντικειμένου
- Χημειοπληροφορική
- Χρηματιστηριακή ανάλυση

Το 2006, η διαδικτυακή εταιρία ταινιών Netflix διοργάνωσε τον πρώτο διαγωνισμό "Βραβείο Netflix" προκειμένου να βρεθεί ένα πρόγραμμα που να προβλέπει καλύτερα τις προβλέψεις των χρηστών και να βελτιώσει την ακρίβεια του αλγορίθμου πρότασης ταινιών Cinematch, που τότε χρησιμοποιούσε, κατά τουλάχιστον 10%. Μια ομάδα αποτελούμενη από ερευνητές από την AT&T Labs-Research σε συνεργασία με τις ομάδες Big Chaos και Pragmatic Theory κατασκεύασε ένα μοντέλο και κέρδισε το βραβείο του 1εκ.δολαρίων το 2009.Λίγο καιρό μετά την απονομή του βραβείου, η διοίκηση της Netflix συνειδητοποίησε ότι τα ποσοστά τηλεθέασης των θεατών δεν ήταν ο καλύτερος δείκτης των μοτίβων τηλεθέασης ("τα πάντα είναι τηλεθέαση")και άλλαξαν αναλόγως την μηχανή σύστασής τους.

Το 2010 το περιοδικό The Wall Street έγραψε ότι η εταιρεία διαχείρισης χρημάτων Rebellion Research's χρησιμοποιούσε την μηχανική μάθηση για να προβλέψει οικονομικές κινήσεις. Το άρθρο περιέγραφε την πρόβλεψη της εταιρείας για την οικονομική κρίση και την οικονομική ανάκαμψη.

Το 2014 έχει αναφερθεί ότι ένας αλγόριθμος μηχανικής μάθησης έχει εφαρμοστεί στην Ιστορία της Τέχνης για την μελέτη πινάκων ζωγραφικής και ότι θα μπορούσε να αποκαλύψει επιρροές μεταξύ καλλιτεχνών που προηγουμένως δεν είχαν εντοπιστεί.

Ηθική

Η Μηχανική Μάθηση, θέτει μια σειρά από ηθικά ζητήματα. Τα συστήματα τα οποία έχουν εκπαιδευτεί σε σύνολα δεδομένων που συλλέγονται με προκαταλήψεις μπορεί να εμφανίζουν αυτές τις προκαταλήψεις κατά τη χρήση, ψηφιοποιώντας πολιτιστικές προκαταλήψεις όπως ο θεσμικός ρατσισμός και ο ταξικός διαχωρισμός. Έτσι η υπεύθυνη συλλογή δεδομένων είναι ένα κρίσιμο κομμάτι της μηχανικής μάθησης.

Ο αλγόριθμος των K-μέσων (K-means)

Ένας από τους πιο γνωστούς αλγόριθμους ομαδοποίησης αυτής της κατηγορίας είναι ο αλγόριθμος των K-μέσων (K-means).

 Ο αριθμός K των ομάδων καθορίζεται πριν την εκτέλεση του αλγορίθμου.

 Ο αλγόριθμος ξεκινά διαλέγοντας K τυχαία σημεία από τα δεδομένα ως τα κέντρα των ομάδων.

 Έπειτα αναθέτει κάθε σημείο στην ομάδα της οποίας το κέντρο είναι πιο κοντά (μικρότερη απόσταση) σε αυτό το σημείο. Στη συνέχεια, υπολογίζει για κάθε ομάδα το μέσο όρο όλων των σημείων της (μέσο διάνυσμα) και ορίζει αυτό ως νέο κέντρο της.

 Τα δύο τελευταία βήματα επαναλαμβάνονται για ένα προκαθορισμένο αριθμό βημάτων ή μέχρι να μην υπάρχει αλλαγή στο διαχωρισμό των σημείων σε ομάδες.

Ο αλγόριθμος των K-μέσων δίνεται σε ψευδογλώσσα στη συνέχεια:

Αλγόριθμος K-μέσων

είσοδος:
    Σύνολο δεδομένων D = {x1, …, xn}
    Αριθμός Ομάδων k
έξοδος:
    Ομάδες Ci
1.//ανάθεση τυχαίων κέντρων
  για i= 1,…,k κάνε:
    θεώρησε mi ως ένα τυχαίο στοιχείο από το D;
2.//ομαδοποίηση
    όσο υπάρχουν αλλαγές στις ομάδες Ci κάνε:
    2 α.//δημιουργία ομάδων
        για i = 1, …, k κάνε
        C i = {x ∈ D| d(mi,x) ≤ d(mj,x) για όλα τα j = 1, …,k, j ≠ k};
    2 β.//υπολογισμός νέων κέντρων
        για i = 1,…,k κάνε
        m i = το μέσο διάνυσμα των σημείων που ανήκουν στην ομάδα Ci;

Παράδειγμα Εκτέλεσης του Αλγορίθμου των K-μέσων

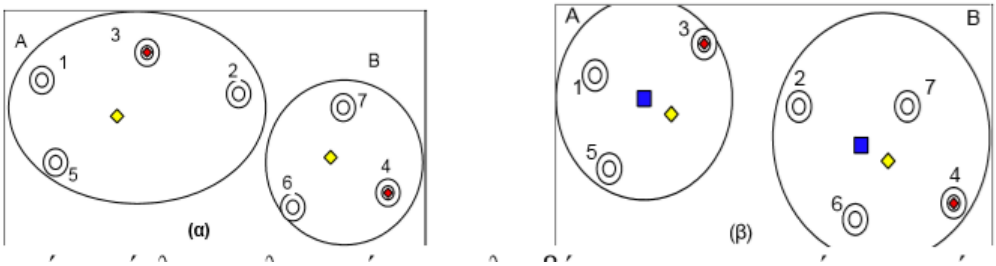
Έστω ότι ο αλγόριθμος εκτελείται με k=2 για τα 7 σημεία του Σχήματος.

 Αρχικά επιλέγονται τυχαία, έστω τα σημεία 3 και 4, ως κέντρα για τις δύο ομάδες Α και Β

 Κάθε σημείο από τα υπόλοιπα ανατίθεται στην ομάδα στης οποίας το κέντρο είναι πιο κοντά, άρα τα σημεία 1, 2, 3 και 5 θα ανήκουν στην ομάδα Α ενώ τα 4, 6 και 7 στην Β.

 Ξαναϋπολογίζονται τα κέντρα κάθε ομάδας και με αυτόν τον τρόπο ολοκληρώνεται ένας κύκλος υπολογισμών.

 Τα νέα κέντρα απεικονίζονται στο Σχήμα (α) με ρόμβο.



 Ο παραπάνω κύκλος υπολογισμών επαναλαμβάνεται χρησιμοποιώντας αυτήν τη φορά τις αποστάσεις των σημείων από τα νέα κέντρα.

• Μια μεταβολή που συντελείται είναι ότι το σημείο 2 αλλάζει ομάδα και ανήκει πλέον στη Β.

• Επιπλέον, τα κέντρα των ομάδων μετατοπίζονται σε νέες θέσεις που στο Σχήμα (β) σημειώνονται με τετράγωνο.

 Κατά την τρίτη επανάληψη των παραπάνω υπολογισμών, δε συντελείται καμία μεταβολή στον πληθυσμό κάθε ομάδας, οπότε η διαδικασία τερματίζει.

Ο αλγόριθμος Apriori

Με την παραδοχή ότι συνήθως έχουμε να κάνουμε με τεράστιες βάσεις δεδομένων, οι οποίες περιέχουν μεγάλο αριθμό από διακριτά αντικείμενα οδηγούμαστε στο συμπέρασμα ότι και τα τυχόν σύνολα που απαρτίζονται από τα αντικείμενα αυτά είναι επίσης μεγάλα.

Για την αποδοτική επίλυση του προβλήματος μας λοιπόν, Agrawal&Srikant (1994) βασίστηκαν σε μια αρχή που ονομάστηκε Apriori ή περιγραφικά αρχή της προς τα κάτω κλειστότητας. Σύμφωνα με αυτή «Ένα κ-σύνολο είναι συχνό μόνο αν όλα τα υποσύνολα του είναι επίσης συχνά».
Αυτή η ιδιότητα οδηγεί στην επίλυση του προβλήματος ξεκινώντας από κάτω, δηλαδή από τα συχνά σύνολα ενός αντικειμένου. Με αυτά ως βάση κατασκευάζουμε τα συχνά σύνολα 2 αντικειμένων, μετά 3 κ.ο. κ. μέχρι να μην μπορούμε να πέσουμε πάνω σε κ-σύνολα που δεν είναι συχνά. Τότε η επαναληπτική διαδικασία σταματά.

Από την διατύπωση του Apriori και μετά εμφανίστηκαν αρκετές παραλλαγές που δημιουργούν βελτιώσεις πάνω στον αλγόριθμο όπως είναι οι τεχνικές hashing (Parketal 1995),τεχνικές partitioning (Savasereet. al. 1995), τεχνικές δειγματοληψίας (Toivonen 1996), σταδιακή εξόρυξη (incrementalmining, Cheunget. al. 1996 και παράλληλη κατανεμημένη εξόρυξη (parallelanddistributingmining, Parket.al. 1995, AgrawalandShafer 1996, Cheunget. al. 1996, Zakiet. al. 1997).

Γενικά οι βελτιώσεις που έγιναν στόχευαν πάντα στον ολοένα και μικρότερο αριθμό περασμάτων της βάσης δεδομένων που όπως θα δούμε είναι το μεγάλο μειονέκτημα του Apriori.

Ο αλγόριθμος Apriori προτάθηκε από τον Rakesh Agrawal το 1994 και είναι ίσως ο κλασικότερος αλγόριθμος ανακάλυψης κανόνων συσχέτισης.

Περιλαμβάνει δυο βασικά βήματα, τη δημιουργία των συχνών συνόλων αντικειμένων και
τη δημιουργία των κανόνων συσχέτισης.

1. Η διαδικασία της δημιουργίας συχνών συνόλων αντικειμένων περιλαμβάνει δύο στάδια:

   I. Αρχικά δημιουργείται ένα σύνολο υποψήφιων συχνών αντικειμένων Ci και στη συνέχεια, χρησιμοποιώντας το όριο υποστήριξης (support), δημιουργείται το σύνολο των συχνών συνόλων αντικειμένων Li.

II. Η διαδικασία επαναλαμβάνεται πραγματοποιώντας διαδοχικά περάσματα στα δεδομένα μέχρι να βρεθούν είτε τα συχνά σύνολα αντικειμένων ενός προκαθορισμένου επιπέδου ή τα μέγιστα συχνά σύνολα αντικειμένων.

Το πρώτο στάδιο επιπλέον αποτελείται από ένα βήμα συνένωσης (join step) και ένα βήμα
κλαδέματος (prune step) τα οποία συνήθως εκτελούνται στη μνήμη και έτσι δεν είναι ιδιαίτερα χρονοβόρα.

2. Για τη δημιουργία των κανόνων συσχέτισης ελέγχεται η εμπιστοσύνη(confidence)

όλων των πιθανών κανόνων που προκύπτουν από τα μέγιστα συχνά σύνολα αντικειμένων και στο τέλος μένουν εκείνοι των οποίων η εμπιστοσύνη ξεπερνά το όριο
που τέθηκε από το χρήστη

Δημιουργία Συχνών Συνόλων Αντικειμένων

Έστω ότι το όριο υποστήριξης είναι sup. Στο πρώτο πέρασμα βρίσκονται τα αντικείμενα εκείνα που εμφανίζονται στη βάση δεδομένων σε ένα ποσοστό sup των καλαθιών ή μεγαλύτερο.

- Αυτό το σύνολο ονομάζεται σύνολο συχνών αντικειμένων (frequent 1-itemset) και συμβολίζεται με L1.
- Από το L1 προκύπτουν, κάνοντας όλους τους δυνατούς συνδυασμούς, τα υποψήφια (συχνά) ζεύγη αντικειμένων C2.

Στο δεύτερο πέρασμα, τα ζεύγη του C2 των οποίων το πλήθος ικανοποιεί το κριτήριο sup, δημιουργούν το L2, δηλαδή τα συχνά ζεύγη (frequent 2-itemsets).

- Από το L2 προκύπτουν οι υποψήφιες (συχνές) τριάδες αντικειμένων C3 που θα χρησιμοποιηθούν στο τρίτο πέρασμα.
- Οι υποψήφιες τριάδες C3 είναι σύνολα του τύπου {A,B,C} τέτοια, ώστε όλα τα υποσύνολά του μεγέθους δυο, δηλαδή τα {A,B}, {A,C}, {B,C}, να ανήκουν στο L2.

Στο τρίτο πέρασμα, υπολογίζεται ο αριθμός των εμφανίσεων των τριάδων του C3 και με βάση το κριτήριο sup δημιουργείται το L3.

Η διαδικασία συνεχίζεται για προκαθορισμένο αριθμό επιπέδων ή μέχρι να αδειάσουν τα υποψήφια συχνά σύνολα αντικειμένων και να δημιουργηθούν τα μέγιστα συχνά σύνολα αντικειμένων.

Παράδειγμα Εφαρμογής του Apriori

Έστω ένα σύνολο δεδομένων που αντιστοιχούν σε 10 διαφορετικά καλάθια αγορών από ένα super market.

- Κάθε καλάθι περιλαμβάνει ένα υποσύνολο των προϊόντων του super market.
- Για παράδειγμα, στο πρώτο καλάθι ο πελάτης αγόρασε μόνο ψωμί και γάλα.

***Πίνακας 6**: Παράδειγμα Apriori*

Καλάθι	Ψωμί	Καφές	Γάλα	Ζάχαρη
#1	1	0	1	0
#2	0	1	0	0
#3	1	0	1	1
#4	0	1	0	1
#5	1	0	1	1
#6	1	1	1	0
#7	1	0	0	1
#8	1	1	1	1
#9	0	0	1	1
#10	1	1	0	1

Έστω επίσης ότι η ζητούμενη υποστήριξη είναι sup=40% και η ζητούμενη εμπιστοσύνη conf=80%.

- Στο πρώτο βήμα, ο Apriori υπολογίζει την υποστήριξη όλων των αντικειμένων, δηλαδή δημιουργεί το σύνολο L1.
  - S{Ψωμί} = 7/10 = 70% ≥ sup
  - S{Καφές} = 5/10 = 50% ≥ sup
  - S{Γάλα} = 6/10 = 60% ≥ sup
  - S{Ζάχαρη} = 7/10 = 70% ≥ sup

  Συνεπώς L1={ Ψωμί, Καφές, Γάλα, Ζάχαρη }

- Στο δεύτερο βήμα, παράγονται όλοι οι δυνατοί συνδυασμοί των αντικειμένων, για να δημιουργηθεί το σύνολο υποψήφιων ζευγών αντικειμένων, δηλαδή το σύνολο C2.
  - C2={ {Ψωμί,Καφές},{Ψωμί,Γάλα},{Ψωμί,Ζάχαρη},{Καφές,Γάλα},
  - {Καφές,Ζάχαρη},{Γάλα,Ζάχαρη} }

  Κατόπιν, υπολογίζεται η υποστήριξη των μελών του C2 και απορρίπτονται εκείνα που δεν ξεπερνούν το όριο ελάχιστης υποστήριξης, ώστε να δημιουργηθεί το σύνολο συχνών ζευγών L2.
  - S({Ψωμί, Καφές}) = 3/10 = 30% < sup (απορρίπτεται)
  - S({Ψωμί, Γάλα}) = 5/10 = 50% ≥ sup
  - κτλ.

  Τελικά: L2={{Ψωμί,Γάλα},{Ψωμί,Ζάχαρη},{Γάλα,Ζάχαρη}}

Από το L2, με τον ίδιο τρόπο, θα δημιουργηθούν τα C3 και L3 (αν τελικά υπάρχουν "συχνές" τριάδες). Στο συγκεκριμένο παράδειγμά, το βήμα δημιουργίας υποψήφιων τριάδων έχει ως εξής:

- Βήμα συνένωσης: {Ψωμί,Γάλα}∪{Ψωμί,Ζάχαρη}= {Ψωμί,Γάλα,Ζάχαρη}
- Βήμα κλαδέματος: Οι επιμέρους δυάδες (ζεύγη) του {Ψωμί,Γάλα,Ζάχαρη} ανήκουν όλες στο L2, άρα:

  - C3={{Ψωμί,Γάλα,Ζάχαρη}}
  - S({Ψωμί,Γάλα,Ζάχαρη})=3/10=30% (απορρίπτεται), άρα L3={}.

Ο αλγόριθμος εύρεσης συχνών συνόλων αντικειμένων σταματά εδώ και συνεπώς το μέγιστο συχνό σύνολο αντικειμένων είναι το L2.

Το επόμενο βήμα είναι η εξαγωγή των κανόνων από τα συχνά σύνολα (στο συγκεκριμένο παράδειγμα μόνο το L2), βάσει της εμπιστοσύνης τους.

- L2={{Ψωμί,Γάλα},{Ψωμί,Ζάχαρη},{Γάλα,Ζάχαρη}}

Ελέγχεται η εμπιστοσύνη όλων των πιθανών κανόνων που μπορεί να προκύψουν από το L2.

- {Ψωμί,Γάλα}
  - Ψωμί→Γάλα: εμπιστοσύνη = 5/7 = 71% < conf (απορρίπτεται)
  - Γάλα→Ψωμί: εμπιστοσύνη = 5/6 = 83% > conf (εγκρίνεται)

- {Ψωμί,Ζάχαρη}
  - Ψωμί→Ζάχαρη: εμπιστοσύνη = 5/7 = 71% < conf (απορρίπτεται)
  - Ζάχαρη→Ψωμί: εμπιστοσύνη = 5/7 = 71% < conf (απορρίπτεται)

- {Γάλα,Ζάχαρη}
  - Γάλα→Ζάχαρη: εμπιστοσύνη = 4/6 = 66% < conf (απορρίπτεται)
  - Ζάχαρη→Γάλα: εμπιστοσύνη = 4/7 = 57% < conf (απορρίπτεται)
  -

Τελικά παράγεται μόνο ο κανόνας: Γάλα→Ψωμί, δηλαδή όποιος αγοράζει Γάλα αγοράζει και Ψωμί.
Αν ελαττώσουμε τη ζητούμενη εμπιστοσύνη στο 70% τότε θα παραχθούν τέσσερις κανόνες.

Μετοχές και Δείκτες Τεχνικής Ανάλυσης

Ανάλυση Μετοχών

Μετοχές είναι η συμμετοχή στην ιδιοκτησία κάποιας εταιρείας. Οι μετοχές είναι οι πλέον εμπορεύσιμες αξίες. Είναι η μονάδα που διαιρείται το μετοχικό κεφάλαιο μίας εταιρείας, παρέχει στον κάτοχο της δικαίωμα ψήφου και δικαίωμα συμμετοχής στα κέρδη που προκύπτουν από τις εργασίες της. Επίσης συμμετέχουν αναλογικά σε νέες εκδόσεις κεφαλαίου (rights issues).

• Η **κοινή μετοχή** είναι ο πιο συνηθισμένος τύπος μετοχής και περιλαμβάνει όλα τα βασικά δικαιώματα ενός μετόχου, όπως δικαίωμα συμμετοχής στα κέρδη, στην έκδοση νέων μετοχών, στο προϊόν της εκκαθάρισης, καθώς και δικαίωμα ψήφου στη Γενική Συνέλευση της εταιρείας και συμμετοχής στη διαχείρισή της.

• Η **προνομιούχος μετοχή** προσφέρει απλά ένα προβάδισμα έναντι των κατόχων κοινών μετοχών, στη λήψη μερίσματος και στη λήψη του προϊόντος της εκκαθάρισης σε περίπτωση διάλυσης της επιχείρησης, αλλά συνήθως στερείται του δικαιώματος ψήφου και συμμετοχής στη διαχείριση της επιχείρησης.

• Η **ονομαστική αξία** της μετοχής προκύπτει κατά την πρώτη έκδοση των μετοχών, διαιρώντας την αξία του μετοχικού κεφαλαίου της εταιρείας με τον αριθμό μετοχών, που εξέδωσε αρχικά. Η ονομαστική αξία μπορεί να αλλάξει μετά από απόφαση της Γενικής Συνέλευσης της εταιρείας.

• Η **λογιστική αξία** της μετοχής αντικατοπτρίζει την πραγματική αξία της και προκύπτει διαιρώντας τα ίδια κεφάλαια της εταιρείας με τον αριθμό μετοχών της εταιρείας σε κυκλοφορία.

•Η **χρηματιστηριακή αξία** της μετοχής, διαμορφώνεται καθημερινά ,μέσω της προσφοράς και της ζήτησης.

<u>Ωφελήματα Μετόχων</u>

Ο επενδυτής βλέπει το χρηματιστήριο ως μια εναλλακτική μορφή τοποθέτησης των χρημάτων που αποταμιεύει, με σκοπό την επιδίωξη ικανοποιητικής απόδοσης. Απόδοση που συνήθως είναι υψηλότερη από αυτήν που προσφέρουν άλλες επενδύσεις όπως οι τραπεζικές καταθέσεις και τα κρατικά ομόλογα.

Μία εταιρεία παρέχει στους εγγεγραμμένους μετόχους της διάφορα ωφελήματα όπως μέρισμα, προμέρισμα, δωρεάν μετοχές (bonus issue), ΔΑΜ (warrants), rights issue κ.α. σε μία συγκεκριμένη ημερομηνία (record day). Οι τίτλοι διαπραγματεύονται χωρίς τα ωφελήματα που παρέχονται οκτώ μέρες πριν την πιο πάνω ημερομηνία, σύμφωνα με τους κανόνες διαπραγμάτευσης του Χρηματιστηρίου. Ένας τίτλος όταν διαπραγματεύεται με ωφέλημα χρησιμοποιείται το πρόθεμα “cum” ενώ χωρίς το σχετικό ωφέλημα χρησιμοποιείται το πρόθεμα “ex”. Δηλαδή εάν πρόκειται για παραχώρηση μερίσματος, ο τίτλος της μετοχής που θα διαπραγματεύεται με μέρισμα θα αναφέρεται ως “cum-dividend” ενώ αν διαπραγματεύεται χωρίς μέρισμα θα αναφέρεται ως “ex –dividend”.

Ερμηνεία δεικτών τεχνικής ανάλυσης

Τεχνική Ανάλυση Μετοχής

Τεχνική ανάλυση μιας μετοχής ή ενός Δείκτη, ή μιας ολόκληρης οργανωμένης αγοράς, είναι η μελέτη της κίνησης των τιμών, όπως αυτή απεικονίζεται σε ένα διάγραμμα τιμής- χρόνου, με στόχο τον χαρακτηρισμό της συμπεριφοράς της και την πρόβλεψη της μελλοντικής της κίνησης. Στηρίζεται στα γραφήματα της Αναλυτικής Γεωμετρίας και της Στατιστικής, τα οποία όμως έχουν εκλαϊκευτεί, για να μπορεί να τα καταλαβαίνει ο μέσος επενδυτής. Η Τεχνική Ανάλυση δεν ασχολείται με τα θεμελιώδη στοιχεία μιας εταιρείας ή μιας αγοράς, όπως είναι οι ισολογισμοί, ο λόγος P/E, ή τα κέρδη, αλλά μόνο με τα γραφήματα των μετοχών (ή των δεικτών).

Βασική υπόθεση της τεχνικής ανάλυσης είναι πως ότι είναι γνωστό για μια εταιρεία είναι ήδη ενσωματωμένο στην τιμή της. Για την τεχνική ανάλυση δεν έχει σημασία τι προκάλεσε την άνοδο ή κάθοδο της τιμής μιας μετοχής, αλλά πόσο θα διαρκέσει αυτή η μεταβολή, είτε προς τα πάνω είτε προς τα κάτω. Επίσης βασική παραδοχή της τεχνικής ανάλυσης αποτελεί το γεγονός ότι η τιμή μιας μετοχής απεικονίζει τα οικονομικά δεδομένα, τις πληροφορίες και τα νέα της και επηρεάζεται σημαντικά από τα ανθρώπινα συναισθήματα και την ψυχολογία που επικρατεί κάθε περίοδο.

Η Τεχνική Ανάλυση μελετάει τις μεταβολές τιμών από το παρελθόν και προσπαθεί να εντοπίσει έγκαιρα τις μεταβολές που θα γίνουν στο μέλλον και κυρίως το σημείο αντιστροφής μιας κίνησης τιμών.

Πρέπει να γίνει κατανοητό σε όποιον δραστηριοποιείται στην αγορά, με εργαλείο την Τεχνική Ανάλυση ότι οι αγορές δεν κινούνται από τα γεγονότα, αλλά από τις προσδοκίες των ανθρώπων. Οι δε τιμές των μετοχών στο ταμπλό του χρηματιστηρίου δεν απεικονίζουν το "σήμερα" αλλά το "αύριο" με ορίζοντα περίπου 3-6 μηνών ή και περισσότερο.

Κάθε αγορά μετοχών, παραγώγων, αξιών ή/και καταναλωτικών ειδών γεννά, με τη συμπεριφορά της, στους εμπλεκόμενους τα απολύτως απαραίτητα ανθρώπινα συναισθήματα της απληστίας, του φόβου, της ανάγκης μιμητισμού των άλλων και ένταξης σε μεγαλύτερα σύνολα, τα οποία αποτελούν τις αιτίες αποτυχημένης δραστηριοποίησης στην αγορά.

Η Τεχνική Ανάλυση, θεωρεί ότι τα ανθρώπινα συναισθήματα απεικονίζονται στους σχηματισμούς των τιμών, και προσπαθεί να απομονώσει τις ψυχολογικές επιδράσεις της αγοράς, εστιάζοντας στα διαγράμματα και τους σχηματισμούς που δημιουργούνται σε αυτά.

Εργαλεία τεχνικής ανάλυσης

Τα εργαλεία της τεχνικής ανάλυσης είναι οι δείκτες. Ένας δείκτης είναι ένας μαθηματικός υπολογισμός που στηρίζεται στην τιμή της μετοχής ή τον όγκο συναλλαγών της ή και στα δύο. Τα ονόματα των κυριοτέρων δεικτών είναι: Κινητός Μέσος Όρος 5 -30 και 200 ημερών, Στοχαστικός δείκτης, Λωρίδες Bollinger κ.λπ. Σήμερα με τη χρήση των υπολογιστών, η τεχνική ανάλυση είναι πολύ πιο εύκολη και χρήσιμη για τον υπολογισμό της κίνησης των τιμών των μετοχών. Επίσης σήμερα υπάρχουν αρκετά προγράμματα Τεχνικής Ανάλυσης όπως είναι το Metastock, τα οποία εφ' όσον εφοδιαστούν με τα ιστορικά στοιχεία μιας μετοχής ή ενός δείκτη, παρουσιάζουν σε γραφήματα όλους τους σημαντικότερος δείκτες Τεχνικής Ανάλυσης

Κυριότερος μεσοπρόθεσμος δείκτης αγορών

Είναι ο λεγόμενος "Κινητός μέσος των 200 ημερών" Θεωρείται ο σημαντικότερος δείκτης για μετοχές, αλλά και για ολόκληρες αγορές. Η κίνηση του έχει διαπιστωθεί ότι επηρεάζεται τόσο από τα οικονομικά στοιχεία, όσο και από τις πολιτικές εξελίξεις (όταν πρόκειται για οργανωμένες αγορές όπως είναι το Χρηματιστήριο Αθηνών), ακόμα και από την γενικότερη ψυχολογία του επενδυτικού κοινού.

Συμπερασματικά

Οποιαδήποτε μετοχή και οποιοσδήποτε δείκτης έχει 50% πιθανότητα να ανέβει και 50% να πέσει. Όποια μέθοδο και όποιο δείκτη και αν χρησιμοποιήσει κανείς δεν μπορεί να προβλέψει την κίνηση των χρηματαγορών με ακρίβεια. Η τεχνική ανάλυση λειτουργεί με πιθανότητες και αποσκοπεί στο να βελτιώσει αυτές τις πιθανότητες προς όφελος του επενδυτή**. Γενικά αν μία μέθοδος πρόβλεψης της κίνησης των τιμών των μετοχών έχει πιθανότητες επιτυχίας 60% θεωρείται επιτυχημένη.**

Ένα σύστημα προβλέψεων λειτουργεί ως εξής : Αρχικά έχουμε τη συλλογή των δεδομένων που αποτελούν τις εισροές του συστήματος. Εν συνεχεία θα πρέπει να γίνει επιλογή της μεθόδου προβλέψεως που θα χρησιμοποιηθεί για την επεξεργασία των δεδομένων μας. **Προκειμένου να επιλέξουμε την καταλληλότερη μέθοδο πρόβλεψης θα πρέπει να λάβουμε υπόψη μας μεταξύ άλλων:** 1) την ακρίβεια των προβλέψεων 2) την ευστάθεια της μεθόδου πρόβλεψης 3) την αντικειμενικότητα στην επεξεργασία των δεδομένων 4) τον απαιτούμενο χρόνο για τη διαμόρφωση προβλέψεων και 5) το κόστος εφαρμογής της μεθόδου.

Επιπροσθέτως εκτός από τα ανωτέρω κριτήρια αξιολόγησης των μεθόδων πρόβλεψης θα πρέπει να λάβουμε υπόψη μας και τυχόν περιορισμούς που υπάρχουν στη λειτουργία ενός συστήματος προβλέψεων. Οι σπουδαιότεροι από αυτούς είναι οι ακόλουθοι : 1) ο διαθέσιμος χρόνος για την προετοιμασία μιας πρόβλεψης 2) η έλλειψη στοιχείων 3) η ποιότητα και η αξιοπιστία των διαθέσιμων στοιχείων 4) η έλλειψη εξειδικευμένου προσωπικού και 5) τα διαθέσιμα μέσα για την επεξεργασία των στοιχείων (υπολογιστές, ειδικά προγράμματα κ.τ.λ). Εφόσον έχουμε επιλέξει την καταλληλότερη μέθοδο προβλέψεων, την εφαρμόζουμε και προβαίνουμε στη διαμόρφωση προβλέψεων που αφορούν τη μελλοντική εξέλιξη των τιμών της υπό εξέταση μεταβλητής.

Τεχνικοί Δείκτες

Πιο ειδικά, με τον όρο τεχνική ανάλυση εννοείται η συστηματική μελέτη και επεξεργασία διαγραμμάτων, με στόχο την όσο δυνατή ασφαλέστερη και ακριβέστερη πρόβλεψη των τιμών των μετοχών.

Ένας δείκτης τεχνικής ανάλυσης είναι ένας μαθηματικός υπολογισμός που έχει να κάνει είτε με την τιμή της μετοχής είτε με τον όγκο συναλλαγών της. Ο κινητός μέσος όρος είναι ένα παράδειγμα δείκτη ανάλυσης. Ουσιαστικά είναι ένας μαθηματικός υπολογισμός σχετικός με την τιμή της μετοχής που χρησιμοποιείται στην προσπάθεια να εκτιμηθούν μελλοντικές αλλαγές στην τιμή της μετοχής. **Η αλλαγή των προσδοκιών των επενδυτών συχνά είναι απότομη και βασίζεται σε ειδήσεις γύρω από την εταιρεία. Όταν η μετοχή είναι σε ανοδική πορεία οι αγοραστές είναι πιο πολλοί από τους πωλητές. Όταν η μετοχή είναι σε καθοδική πορεία αυτοί που πουλάνε είναι πιο πολλοί από αυτούς που αγοράζουν.**

Όταν γίνεται λόγος για την κατεύθυνση μιας μετοχής εννοείται μια σταθερή πορεία αλλαγής της τιμής της μετοχής προς τα πάνω ή προς τα κάτω (που είναι συνδεδεμένη με τις προσδοκίες των επενδυτών). Η κατεύθυνση μιας μετοχής διαφέρει από τα επίπεδα στήριξης ή αντίστασης, γιατί κατεύθυνση σημαίνει αλλαγή, ενώ αντίσταση ή στήριξη σημαίνει εμπόδιο στην αλλαγή.

Η τεχνική ανάλυση επιδιώκει να εντοπίζει έγκαιρα την έναρξη τάσης των τιμών και να εκμεταλλεύεται την τάση αυτή για την επίτευξη κέρδους. Οι πιο κερδοφόρες αγοραπωλησίες πραγματοποιούνται μέσα σε τάση, διότι η κατεύθυνση της τιμής είναι συγκεκριμένη και προβλέψιμη με μεγάλη πιθανότητα επιτυχίας.

Για το λόγο αυτό έχουν αναπτυχθεί ειδικοί τεχνικοί δείκτες που δείχνουν:

1.Το ξεκίνημα μιας νέας τάσης των τιμών (ανοδική ή καθοδική)

2.Την ταχύτητα και την ορμή αυτής της τάσης

3.Πόσο «ώριμη» είναι η τάση και πότε έχουμε τερματισμό της τάσης

4.Εάν η τάση πρόκειται να αναστραφεί.

Γνωρίζοντας αυτούς τους τέσσερις βασικούς παράγοντες της τάσης, μπορεί να καθοριστεί επακριβώς πότε θα αγοραστεί και θα πουληθεί τη μετοχή ώστε να τηρείται πιστά ο κανόνας «αγόραζε φθηνά, πούλα ακριβά».

Οι τεχνικοί δείκτες που μετρούν αυτούς τους τέσσερις παράγοντες και δίνουν μ ια ολοκληρωμένη η εικόνα της τάσης των τιμών, ονομάζονται δείκτες τάσης (trend-following indicators). Σύμφωνα με την αγγλική ονομασία τους: οι δείκτες αυτοί «ακολουθούν» την τάση. Αυτό στην πράξη σημαίνει πως οι δείκτες αυτοί δουλεύουν σωστά μόνο όταν οι τιμές των μετοχών είναι μέσα σε περιβάλλον τάσης. Επομένως, δε λειτουργούν σωστά όταν βρίσκονται σε ατασικές-πλευρικές φάσεις των τιμών, δηλαδή μέσα σε ζώνες συναλλαγών ή σχηματισμούς τιμών (τρίγωνα, σφήνες κ.λπ.)

Κινητός Μέσος Όρος

Γενικά

Ο απλούστερος και δημοφιλέστερος δείκτης τάσης των τιμών είναι ο κινητός μέσος. Πολλοί άλλοι σημαντικοί δείκτες τάσης βασίζονται στην αρχή λειτουργίας του κινητού μέσου. Με τον κινητό μέσο είναι δυνατό να απαλειφτούν έντονες καθημερινές διακυμάνσεις της τιμής της μετοχής. Με αυτό τον τρόπο ο δείκτης δίνει την ομαλοποιημένη τάση (smoothed trend) της μετοχής.

Ο κινητός μέσος δίνει ακριβή σήματα αγοράς και πώλησης, καθώς η τάση της τιμής ξεκινάει, εξελίσσεται και ωριμάζει. Επίσης με τη χρήση του μπορεί να αφαιρεθεί η τάση από τις τιμές (detrending) και να μελετηθεί η κυκλική συμπεριφορά της μετοχής.

Επίσης ο κινητός μέσος έχει και μια άλλη ερμηνεία που αφορά περισσότερο τη ψυχολογία των επενδυτών. Είναι ένας δείκτης που μας δίνει τη μέση τιμή κτήσης της μετοχής των τελευταίων ημερών. Όταν ο η τιμή κλεισίματος βρίσκεται πάνω από τον κινητό μέσο, η μέση τιμή κτήσης της μετοχής είναι χαμηλότερη από την τρέχουσα και οι επενδυτές (τουλάχιστον αυτοί που τοποθετήθηκαν πρόσφατα) διατηρούν κλίμα ευφορίας και αισιοδοξίας από την άνοδο των κερδών τους. Άρα η αγορά δείχνει σθένος. Το αντίθετο συμβαίνει όταν η τρέχουσα τιμή κλεισίματος βρίσκεται χαμηλότερα από τον κινητό μέσο, δηλαδή από τη πρόσφατη μέση τιμή κτήσης της μετοχής. Το κλίμα είναι σαφώς αρνητικό με τους επενδυτές να υφίστανται ζημίες κατά μέσο όρο.

Συνεπώς συγκεκριμένοι κανόνες συναλλαγών που προκύπτουν από τη χρήση του κινητού μέσου, οι οποίοι είναι οι εξής:

1. Η ανοδική διάσπαση του ΚΜ από την τιμή κλεισίματος δίνει σήμα ενάρξεως ανοδικής τάσης και αγοράς της μετοχής. Στην περίπτωση ανοδικής τάσης, ο ΚΜ παρέχει στήριξη στην τιμή όταν αυτή επιχειρεί να κάνει διορθωτικές πτώσεις.

2. Η πτωτική διάσπαση του ΚΜ από την τιμή κλεισίματος σηματοδοτεί έναρξη πτωτικής τάσης και μας δίνει το σήμα πώλησης της μετοχής. Στην πτωτική τάση, ο ΚΜ παρέχει το επίπεδο αντίστασης της τιμής, σε κάθε περίπτωση ανοδικής αντίδρασης της.

3. Η μεταβολή της κλίσης του ΚΜ επιβεβαιώνει την αναστροφή της τάσης. Δηλαδή, όταν ένας ανοδικός ΚΜ αντιστρέφεται και αποκτά αρνητική κλίση, επιβεβαιώνει το γεγονός πως η προϋπάρχουσα ανοδική τάση έχει ήδη αντιστραφεί σε πτωτική. Το σήμα αυτό λειτουργεί μόνο ως επιβεβαίωση του γεγονότος αντιστροφής, διότι έρχεται καθυστερημένα λόγω της χρονικής υστέρησης του ΚΜ.

4. Όσο μεγαλύτερης ο αριθμός των ημερών στον ΚΜ, τόσο πιο μακροχρόνια είναι η τάση που αναλύει και τόσο πιο αξιόπιστα είναι τα σήματα αγοραπωλησιών που μας δίνει.

5. Ο ΚΜ δεν προβλέπει την τάση, απλά ακολουθεί την τάση. Πρόκειται για ένα είδος «κινητής» γραμμής τάσης.

Αποκλίσεις

Όταν ο δείκτης Σύγκλισης-Απόκλισης αποκλίνει σε πορεία από την τιμή της μετοχής, αυτό είναι ένδειξη ότι η τωρινή κατεύθυνση της μετοχής είναι πιθανό να αντιστραφεί. Μια αρνητική απόκλιση (bearish divergence) συμβαίνει όταν ο δείκτης πέφτει όλο και πιο χαμηλά, ενώ οι τιμές δεν ακολουθούν την ίδια πορεία. Μια θετική απόκλιση (bullish divergence) συμβαίνει όταν ο δείκτης ανεβαίνει όλο και πιο ψηλά, ενώ η μετοχή όχι. Και οι δύο αυτές αποκλίσεις θεωρούνται πιο σημαντικές όταν συμβαίνουν σε επίπεδα υπερ-αγοράς ή υπερ-πώλησης.

Υπάρχουν διάφοροι τύποι δεικτών «ΚΙΝΗΤΟΥ ΜΕΣΟΥ ΟΡΟΥ» (μαθηματικές συναρτήσεις για των υπολογισμό των μελλοντικών τιμών μετοχών) όπως:

• ΑΠΛΟΣ ΚΙΝΗΤΟΣ ΜΕΣΟΣ ΟΡΟΣ (Simple moving average)

Ο απλός κινητός μέσος όρος είναι ένας αριθμητικός μέσος όρος δεδομένων τιμών. Υπολογίζεται προσθέτοντας την τιμή κάθε διαστήματος και διαιρώντας το σύνολο με τον αριθμό των διαστημάτων που καλύπτει ο κινητός μέσος όρος. Ο μέσος αυτός αποκαλείται «κινητός» γιατί κάθε φορά που γίνεται διαθέσιμη μια νέα παρατήρηση, μπορεί να υπολογιστεί και να χρησιμοποιηθεί ως πρόβλεψη ένας νέος αριθμητικός μέσος. Για παράδειγμα, για να υπολογίσουμε τον κινητό μέσο όρο 25 ημερών προσθέτουμε τις τιμές κλεισίματος των τελευταίων 25 ημερών ενός τίτλου και στη συνέχεια διαιρούμε διά 25.

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

$$SMA_{\text{today}} = SMA_{\text{yesterday}} - \frac{p_{M-n}}{n} + \frac{p_M}{n}$$

Οι προβλέψεις μιας χρονοσειράς Yt, όπου t=1,2,…,n, χρησιμοποιώντας τη μέθοδο του απλού κινητού μέσου, δίνονται από τον ακόλουθο μαθηματικό τύπο :

$$\hat{Y}t+1 = Mt+1 = 1 / m * (\Sigma jYt-j+1) \text{ , όπου } j=1,2,\ldots,m$$

Όπου :

$\hat{Y}t+1$ = η πρόβλεψη για την περίοδο (t+1)
Mt+1 = ο απλός κινητός μέσος για την περίοδο (t+1) και
m = ο αριθμός των περιόδων που χρησιμοποιούνται για τον υπολογισμό του απλού κινητού μέσου

Με τη μέθοδο του απλού κινητού μέσου μπορούμε να εξομαλύνουμε τις τιμές της χρονοσειράς. Αναλυτικότερα, εφαρμόζοντας την μαθηματική σχέση που αναπτύξαμε παραπάνω στις παρατηρήσεις της χρονοσειράς εξομαλύνουμε τις τελευταίες (n – m) παρατηρήσεις της χρονοσειράς. Οι τιμές που προκύπτουν χρησιμοποιούνται ως προβλέψεις των αντίστοιχων περιόδων και εν συνεχεία, βάσει αυτών, προσδιορίζουμε τις τιμές των σφαλμάτων της πρόβλεψης. Ένας εναλλακτικός τρόπος για τη δημιουργία προβλέψεων, στην περίπτωση που η τιμή του m είναι γνωστή, είναι ο εξής :

$$\hat{Y}t+1 = \hat{Y}t + Yt / m – Yt-m / m$$

Ο ανωτέρω μηχανισμός μας δείχνει τον τρόπο με τον οποίο η μέθοδος του απλού κινητού μέσου αναπροσαρμόζει τις προβλέψεις της χρονοσειράς κάθε φορά που μια νέα παρατήρηση γίνεται διαθέσιμη.

Οι κυριότερες αντιρρήσεις που έχουν εκφραστεί αναφορικά με τη χρησιμοποίηση της μεθόδου του απλού κινητού μέσου είναι οι εξής :

- Η ίση στάθμιση των δεδομένων κάθε μιας εκ των προηγούμενων περιόδων μπορεί να μην εκφράζει ικανοποιητικά την παρούσα κατάσταση.

- Η αύξηση του αριθμού των καλυπτόμενων χρονικών περιόδων n, έχει ως αποτέλεσμα την μεγαλύτερη εξομάλυνση των διακυμάνσεων, αλλά κάνει τη μέθοδο λιγότερο ευαίσθητη στις πραγματικές αλλαγές των δεδομένων.
- Ο απλός κινητός μέσος παρουσιάζει μία χρονική υστέρηση σε αλλαγές των πραγματικών τιμών των δεδομένων λόγω μακροχρόνιας τάσεως ή άλλων αιτιών.
- Η μέθοδος του απλού κινητού μέσου απαιτεί εκτεταμένα αρχεία για δεδομένα παρελθουσών περιόδων.
- Με τη μέθοδο του απλού κινητού μέσου είναι δυνατή η πρόβλεψη μόνο για την επόμενη χρονική περίοδο.
-

• ΑΘΡΟΙΣΤΙΚΟΣ ΚΙΝΗΤΟΣ ΜΕΣΟΣ ΟΡΟΣ (Cumulative moving average)

Ο αθροιστικός κινητός μέσος είναι και αυτός ένας αριθμητικός μέσος όρος δεδομένων τιμών. Υπολογίζεται προσθέτοντας τις τιμές κάθε διαστήματος και διαιρώντας το σύνολο με τον αριθμό των διαστημάτων που καλύπτει ο κινητός μέσος όρος.

$$CA_i = \frac{x_1 + \cdots + x_i}{i}.$$

$CA_{i-1} = \frac{x_{i+1} + iCA_i}{i+1}$; όπου το CA_0 μπορεί να θεωρηθεί ίσο με το 0.

• ΣΤΑΘΜΙΣΜΕΝΟΣ ΚΙΝΗΤΟΣ ΜΕΣΟΣ ΟΡΟΣ (Weighted moving average)

Η μέθοδος του σταθμισμένου κινητού μέσου είναι, όπως και αυτή του απλού κινητού μέσου, μια πολύ απλή μέθοδος προβλέψεων. Μια από τις αντιρρήσεις που έχουν εκφραστεί για τον απλό κινητό μέσο είναι η ίση στάθμιση των δεδομένων κάθε μιας εκ των προηγούμενων περιόδων, η οποία μπορεί να μην εκφράζει ικανοποιητικά την παρούσα κατάσταση. Το μειονέκτημα αυτό μπορούμε να το εξουδετερώσουμε χρησιμοποιώντας το σταθμισμένο κινητό μέσο. Η λογική και η μεθοδολογία της μεθόδου αυτής είναι ακριβώς η ίδια με αυτή του απλού κινητού μέσου με τη μόνη διαφορά ότι εδώ σταθμίζουμε τα δεδομένα των περιόδων που εξετάζουμε, χρησιμοποιώντας διαφορετικούς συντελεστές βαρύτητας-στάθμισης ανάλογα με το χρονικό ορίζοντα απόκτησης των δεδομένων. Συνήθως, τα δεδομένα των πιο πρόσφατων περιόδων τα σταθμίζουμε χρησιμοποιώντας συντελεστές βαρύτητας που είναι μεγαλύτεροι από αυτούς που χρησιμοποιούνται για τη στάθμιση δεδομένων πιο μακρινών περιόδων. Το άθροισμα όλων των συντελεστών βαρύτητας ισούται με τη μονάδα, κάτι που ισχύει και στην περίπτωση του απλού κινητού μέσου.

Η επιλογή των συντελεστών βαρύτητας εξαρτάται από την κρίση και την εμπειρία του εκάστοτε αναλυτή. Γενικά θα μπορούσαμε να πούμε ότι αν το πρόσφατο, σε σχέση με το απώτερο, παρελθόν θεωρείται καλύτερος δείκτης πρόβλεψης του μέλλοντος, τότε μεγαλύτεροι συντελεστές βαρύτητας θα πρέπει να δίνονται στις πιο πρόσφατες παρατηρήσεις. Ωστόσο, αν η χρονολογική σειρά που εξετάζουμε διακρίνεται από μεγάλη μεταβλητότητα, τότε ίσως είναι ορθότερο να χρησιμοποιούμε ίσους συντελεστές βαρύτητας για κάθε παρατήρηση.

Ο τύπος που μας δίνει τον σταθμισμένο κινητό μέσο είναι ο ακόλουθος:

Σταθμισμένος Κινητός Μέσος =
Σ (συντελεστής βαρύτητας για την περίοδο t)*(παρατήρηση της περιόδου t)/
Σ (συντελεστές βαρύτητας)

Ή με την μαθηματική του μορφή:

$$\mathrm{WMA}_M = \frac{np_M + (n-1)p_{M-1} + \cdots + 2p_{(M-n+2)} + p_{(M-n+1)}}{n + (n-1) + \cdots + 2 + 1}$$

Αξίζει να σημειωθεί ότι τα μειονεκτήματα που αναφέραμε στην προηγούμενη ενότητα και που αφορούν τον απλό κινητό μέσο, ισχύουν, με εξαίρεση αυτό της ίσης στάθμισης, και για τον σταθμισμένο κινητό μέσο.

• ΕΚΘΕΤΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ (Exponential moving average)

Στα πλαίσια της παρουσίασης του απλού κινητού μέσου, που αποτελεί μια πολύ απλή μέθοδο προβλέψεων, διαπιστώσαμε ότι ένα από τα μειονεκτήματα της μεθόδου είναι η ίση βαρύτητα που δίνει, κατά τον υπολογισμό των προβλέψεων, σε κάθε παρατήρηση ανεξάρτητα από τόσο κοντά ή μακριά βρίσκεται σε σχέση με την προβλεπόμενη περίοδο. Ένας τρόπος για την αντιμετώπιση αυτού του προβλήματος είναι η χρήση της μεθόδου του σταθμισμένου κινητού μέσου. Μία άλλη μέθοδος που μπορεί να αντιμετωπίσει το ανωτέρω πρόβλημα και που τυγχάνει ευρείας αποδοχής σε διάφορες περιπτώσεις που απαιτούνται προβλέψεις οικονομικών μεγεθών, είναι η μέθοδος της απλής εκθετικής εξομάλυνσης.

Σύμφωνα με τη μέθοδο αυτή οι προβλέψεις δημιουργούνται βάσει κάποιου σταθμικού μέσου όρου, έτσι ώστε να δίνεται διαφορετική βαρύτητα σε κάθε παρατήρηση. Αναλυτικότερα, με τη μέθοδο αυτή δίνεται μεγαλύτερη βαρύτητα στις πιο πρόσφατες παρατηρήσεις, σε σχέση πάντοτε με την προβλεπόμενη περίοδο, από αυτή που δίνεται στις πιο απομακρυσμένες. Για να μπορέσουμε να κατανοήσουμε τον τρόπο λειτουργίας της μεθόδου αυτής θα θεωρήσουμε ότι οι προβλέψεις της χρονοσειράς που εξετάζουμε δημιουργούνται ως εξής :

$$\hat{Y}t+1 = \alpha Yt + \alpha(1-\alpha)Yt-1+\alpha(1-\alpha)2Yt-2+\ldots\ldots..$$

Τη σχέση αυτή θα την ονομάσουμε σχέση (1). Η παράμετρος α της σχέσης αυτής ονομάζεται σταθερά εξομάλυνσης (smoothing constant) και λαμβάνει τιμές από μηδέν έως και ένα, δηλαδή $0 \le \alpha \le 1$. Από τη σχέση (1) γίνεται αντιληπτό ότι η πρόβλεψη $\hat{Y}t+1$ προκύπτει ως ένας σταθμικός μέσος όρος των παρατηρήσεων της χρονοσειράς, καθώς το άθροισμα όλων των συντελεστών βαρύτητας ισούται με τη μονάδα. Επιπροσθέτως, διαπιστώνουμε ότι όσο αυξάνεται ο αριθμός των χρονικών περιόδων μεταξύ της πρόβλεψης και της πραγματικής τιμής της χρονοσειράς τόσο οι συντελεστές βαρύτητας μειώνονται και μάλιστα εκθετικά. Γι' αυτό και η υπό εξέταση μέθοδος καλείται απλή «εκθετική» εξομάλυνση. Γίνεται λοιπόν κατανοητό, ότι όσο μεγαλύτερη είναι η τιμή της

σταθεράς εξομάλυνσης α τόσο πιο μεγάλη είναι η βαρύτητα που δίνεται στις πιο πρόσφατες παρατηρήσεις και τόσο πιο μικρή ή και μηδαμινή είναι η βαρύτητα που δίνεται στις πιο απομακρυσμένες παρατηρήσεις.

Με βάση τη σχέση (1) η πρόβλεψη για την περίοδο t, Ŷt , που γίνεται στην αρχή της περιόδου αυτής, θα είναι :

$$\hat{Y}t= \alpha Yt-1 + \alpha(1-\alpha)Yt-2+\alpha(1-\alpha)2Yt-3+.....$$

Τη σχέση αυτή θα την ονομάσουμε σχέση (2). Αν πολλαπλασιάσουμε και τα δύο μέλη της σχέσης (2) με (1-α) και εν συνεχεία την αφαιρέσουμε από τη σχέση (1), τότε προκύπτει η εξής σχέση :

$$\hat{Y}t+1= \alpha Yt+ (1-\alpha)\hat{Y}t$$

Η ανωτέρω σχέση αποτελεί την μαθηματική έκφραση της μεθόδου της απλής εκθετικής εξομάλυνσης για t = 2,3,….,n και με αρχική συνθήκη Ŷ2=Y1.

Γίνεται σαφές ότι η πρόβλεψη Ŷt+1 της περιόδου (t+1) είναι ένας σταθμικός μέσος όρος της πραγματικής τιμής Yt και της προβλεπόμενης τιμής Ŷt της περιόδου t με συντελεστές βαρύτητας α και (1-α) αντίστοιχα. Αυτό σημαίνει ότι αν η τιμή της σταθεράς εξομάλυνσης α είναι α=0,3, τότε η πρόβλεψη της επόμενης περιόδου προσδιορίζεται κατά 30% από την πραγματική τιμή και κατά 70% από την προβλεπόμενη της τρέχουσας περιόδου. Να σημειωθεί πως όταν η τιμή της σταθεράς εξομάλυνσης λαμβάνει τις ακραίες τιμές ένα και μηδέν τότε : για α=1 η πρόβλεψη της επόμενης περιόδου (t+1) είναι η πραγματική τιμή της τρέχουσας περιόδου t και για α=0 η πρόβλεψη της επόμενης περιόδου (t+1) είναι ίση με την πρόβλεψη της τρέχουσας περιόδου t.

Παρατηρούμε ότι όσο πιο μικρές είναι οι τιμές της σταθεράς α τόσο περισσότερο εξομαλύνονται οι παρατηρήσεις της χρονοσειράς και όσο πιο μεγάλες είναι οι τιμές της σταθεράς α τόσο πιο γρήγορα αντιδρά η εν λόγω μέθοδος στις πραγματικές μεταβολές των παρατηρήσεων της χρονοσειράς.

Συνεπώς τίθεται ένα ερώτημα: Μεγαλύτερη εξομάλυνση των παρατηρήσεων της χρονοσειράς και άρα μικρή τιμή για το α ή μεγαλύτερη ανταπόκριση στις πραγματικές μεταβολές των παρατηρήσεων και άρα μεγάλη τιμή για το α ; Η απάντηση στο ερώτημα αυτό δεν είναι καθόλου εύκολη. Θα λέγαμε ότι σπουδαίο ρόλο διαδραματίζει η κρίση του ερευνητή και η τυχόν προηγούμενη εμπειρία που έχει για τη συγκεκριμένη χρονολογική σειρά. Βέβαια, το πιο σωστό είναι η «άριστη» τιμή για το α να προσδιορίζεται από τα δεδομένα της χρονοσειράς. Συγκεκριμένα από τις τιμές του α, $0 \leq \alpha \leq 1$, επιλέγουμε εκείνη που ελαχιστοποιεί την τιμή του κριτηρίου MSE ή κάποιου άλλου κριτηρίου.

Αυτό μπορεί να γίνει εύκολα και γρήγορα με τη χρήση ενός σύγχρονου υπολογιστικού πακέτου. Ωστόσο, να σημειωθεί πως η «άριστη» τιμή του α προσδιορίζεται βάσει ενός συγκεκριμένου αριθμού παρατηρήσεων. Έτσι όταν ο αριθμός αυτός μεταβάλλεται καλό θα είναι να προβαίνουμε σε επανεκτίμηση της τιμής της σταθεράς α, ώστε οι προβλέψεις που προκύπτουν να είναι όσο το δυνατόν πιο αξιόπιστες.

Η μαθηματική σχέση που εκφράζει τη μέθοδο της απλής εκθετικής εξομάλυνσης μπορεί να λάβει και την ακόλουθη μορφή :

$$\hat{Y}t+1= \hat{Y}t + \alpha(Yt - \hat{Y}t) = \hat{Y}t + \alpha et$$

Η ανωτέρω σχέση μας δείχνει ότι η πρόβλεψη της περιόδου (t+1) είναι ίση με την πρόβλεψη της περιόδου t συν το σφάλμα της πρόβλεψης et, πολλαπλασιασμένο με την τιμή της σταθεράς εξομάλυνσης α. Κατά συνέπεια όσο μεγαλύτερη είναι η τιμή του α τόσο πιο μεγάλη βαρύτητα δίνεται στο σφάλμα της πρόβλεψης.

Τα κυριότερα πλεονεκτήματα της μεθόδου που την κάνουν ευρέως χρησιμοποιούμενη είναι:

1. η μεγάλη ακρίβεια πρόβλεψης,
2. η ευκολία υπολογισμού,
3. η απαίτηση ελάχιστων δεδομένων για τον υπολογισμό.

Ο Γενικός τύπος υπολογισμού του εκθετικού μέσου όρου είναι:

$$S_1 = Y_1$$

$$\text{Για } t > 1, \quad S_t = \alpha \cdot Y_t + (1-\alpha) \cdot S_{t-1}$$

$$\mathrm{EMA}_{\mathrm{today}} = \mathrm{EMA}_{\mathrm{yesterday}} + \alpha \times (\mathrm{price}_{\mathrm{today}} - \mathrm{EMA}_{\mathrm{yesterday}})$$

Σημείωση: Όσο μικρότερη η τιμή του α τόσο μεγαλύτερη εμπιστοσύνη δείχνουμε στην αρχική μας πρόβλεψη.

Συνήθεις τιμές του α: 0.1 – 0.3

Κριτήρια για την αξιολόγηση της ακρίβειας των προβλέψεων

Στην ενότητα αυτή θα παρουσιάσουμε τα πιο γνωστά κριτήρια που υπάρχουν για την αξιολόγηση των μεθόδων προβλέψεων. Σκοπός της εφαρμογής των κριτηρίων αυτών είναι **η επιλογή της πιο κατάλληλης και αξιόπιστης μεθόδου πρόβλεψης**. Τα κριτήρια αυτά βασίζονται, στις αποκλίσεις που παρουσιάζονται ανάμεσα στις προβλεπόμενες και στις πραγματικές τιμές της χρονοσειράς. Όσο μικρότερες είναι οι αποκλίσεις αυτές τόσο πιο αξιόπιστη θεωρείται η μέθοδος πρόβλεψης που εφαρμόστηκε. Η απόκλιση ανάμεσα στην προβλεπόμενη και την πραγματική τιμή για μια περίοδο t όπου t = 1,2,…..,n καλείται «Σφάλμα Πρόβλεψης» και ορίζεται ως εξής :

$$et = Yt – \hat{Y}t$$

Όπου : Yt = πραγματική τιμή της περιόδου t και
`Ŷt = προβλεπόμενη τιμή της περιόδου t

Η ανωτέρω σχέση εκφράζει για κάθε περίοδο t τη διαφορά μεταξύ της πραγματικής τιμής (Yt) και της αντίστοιχης προβλεπόμενης τιμής (Ŷt) που προήλθε από τη μέθοδο πρόβλεψης που εφαρμόστηκε. Γίνεται λοιπόν κατανοητό ότι για να προσδιορίσουμε την αξιοπιστία μιας μεθόδου πρόβλεψης θα πρέπει να μελετήσουμε τη διαχρονική συμπεριφορά που παρουσιάζουν οι τιμές των σφαλμάτων της πρόβλεψης.

Αξίζει να σημειωθεί ότι τα κριτήρια που θα εξετάσουμε στη συνέχεια μπορούν να χρησιμοποιηθούν όχι μόνο για την αξιολόγηση μιας μεθόδου πρόβλεψης, αλλά και για την επιλογή της καλύτερης μεταξύ δύο ή και περισσοτέρων εναλλακτικών μεθόδων προβλέψεων.

ΜΕΣΟ ΣΦΑΛΜΑ (MEAN ERROR – ME)

Ένα πολύ απλό μέτρο για την αξιολόγηση της ακρίβειας μιας μεθόδου πρόβλεψης είναι το μέσο σφάλμα. Το μέσο σφάλμα ορίζεται ως εξής : είναι το άθροισμα των τιμών του σφάλματος της πρόβλεψης διαιρούμενο με τον αριθμό των περιόδων n στις οποίες έγιναν προβλέψεις. Με άλλα λόγια:

ΜΕ = 1 / n * (Σtet) , Όπου t =1,2,…,n

Η μονάδα μέτρησης του κριτηρίου αυτού είναι η ίδια με εκείνη των τιμών της χρονοσειράς και έτσι η ερμηνεία του είναι εύκολη. Το σοβαρό μειονέκτημα όμως που παρουσιάζει είναι ότι η τιμή που λαμβάνει επηρεάζεται από τις θετικές και αρνητικές τιμές του σφάλματος. Για να αποφύγουμε το πρόβλημα αυτό μπορούμε να καταφύγουμε στο επόμενο κριτήριο που είναι η μέση απόλυτη απόκλιση.

ΜΕΣΟ ΑΠΟΛΥΤΟ ΣΦΑΛΜΑ (MEAN ABSOLUTE ERROR).

Είναι το σφάλμα ως προς τον μέσο όρο των απολύτων τιμών της απόκλισης των πραγματικών τιμών από τις προβλεπόμενες τιμές. Το σφάλμα αυτό δίνεται από τον τύπο:

$$\mathrm{MAE} - \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| - \frac{1}{n}\sum_{i=1}^{n}|e_i|\,.$$

Σημείωση: Οι θετικές αποκλίσεις εξουδετερώνονται από τις αρνητικές, έτσι ώστε να μπορεί να εμφανιστεί τελικά μικρό μέσο σφάλμα αν και έχουν σημειωθεί στην πραγματικότητα πολύ μεγάλες (θετικές και αρνητικές) αποκλίσεις.

ΜΕΣΗ ΑΠΟΛΥΤΗ ΑΠΟΚΛΙΣΗ (MEAN ABSOLUTE DEVIATION –MAD)

Πρόκειται για ένα απλό και εύχρηστο κριτήριο για την αξιολόγηση της ακρίβειας μιας μεθόδου πρόβλεψης. Η μέση απόλυτη απόκλιση εκφράζει την μέση τιμή των απόλυτων αποκλίσεων μεταξύ των προβλεπόμενων και πραγματικών τιμών μιας χρονοσειράς και ορίζεται ως το άθροισμα των απόλυτων τιμών του σφάλματος της πρόβλεψης ως προς τον αριθμό των περιόδων n για τις οποίες έγιναν προβλέψεις. Δηλαδή :

$$\mathrm{MAD} = \mathrm{median}_i\left(\,|X_i - \mathrm{median}_j(X_j)|\,\right),$$

ή

MAD = 1 / n * (Σt letI), Όπου t =1,2,…,n

Πχ ας εξετάσουμε τα δεδομένα (1, 1, 2, 2, 4, 6, 9). Έχει μια διάμεση τιμή 2. Οι απόλυτες αποκλίσεις περίπου για το 2 είναι (1, 1, 0, 0, 2, 4, 7) οι οποίες με τη σειρά τους έχουν μία διάμεση τιμή 1 (επειδή οι απόλυτες ταξινόμηση αποκλίσεις είναι (0, 0, 1, 1, 2, 4, 7). Έτσι, η μέση απόλυτη απόκλιση για αυτά τα δεδομένα είναι 1.

Στα θετικά του κριτηρίου καταλογίζονται : α) το γεγονός ότι η μονάδα μέτρησης του, όπως και στην περίπτωση του μέσου σφάλματος, είναι η ίδια με εκείνη των τιμών της χρονοσειράς και έτσι η ερμηνεία του είναι εύκολη β) επειδή χρησιμοποιούμε τις απόλυτες τιμές του σφάλματος της πρόβλεψης, το MAD είναι ανεξάρτητο από το εάν οι τιμές των προβλέψεων είναι μικρότερες (υποεκτίμηση) ή μεγαλύτερες (υπερεκτίμηση) των πραγματικών τιμών και γ) η σοβαρότητα του σφάλματος, δηλαδή το κόστος που δημιουργείται από το σφάλμα της πρόβλεψης, σχετίζεται γραμμικά με το μέγεθος του σφάλματος και κατά συνέπεια η σοβαρότητα του σφάλματος είναι η ίδια, είτε προέρχεται από λίγα και μεγάλα σφάλματα είτε από πολλά και μικρά σφάλματα, που έχουν το ίδιο συνολικό άθροισμα απόλυτων τιμών.

Η ΜΕΣΗ ΑΠΟΛΥΤΗ ΠΟΣΟΣΤΙΑΙΑ ΑΠΟΚΛΙΣΗ (Percent Absolute Deviation, PMAD) δίνεται από τον τύπο:

$$PMAD = \frac{\sum_{t=1}^{N} |E_t|}{\sum_{t=1}^{N} |Y_t|}$$

ΜΕΣΟ ΣΦΑΛΜΑ ΤΕΤΡΑΓΩΝΟΥ (MEAN SQUARED ERROR – MSE)

Μια άλλη τεχνική που είναι γενικά παραδεκτή για την αξιολόγηση των μεθόδων της εκθετικής εξομάλυνσης (καθώς και άλλων) είναι το μέσο σφάλμα τετραγώνου. Το μέσο σφάλμα τετραγώνου είναι η μέση τιμή των τετραγώνων των αποκλίσεων μεταξύ των προβλεπόμενων και των πραγματικών τιμών μιας χρονοσειράς και ορίζεται ως το άθροισμα των τετραγώνων των σφαλμάτων ως προς τον αριθμό των χρονικών περιόδων n στις οποίες έγιναν προβλέψεις. Δηλαδή :

ή

MSE = 1 / n * (Σt (et)2) , Όπου t =1,2,…,n

Ορισμένες ουσιώδεις παρατηρήσεις που μπορούν να γίνουν σχετικά με το υπό εξέταση κριτήριο είναι οι ακόλουθες : α) η μονάδα μέτρησης εις την οποία είναι εκφρασμένο το MSE είναι η ίδια με αυτή των τιμών της χρονοσειράς αλλά υψωμένη στο τετράγωνο. Το γεγονός αυτό δυσχεραίνει σημαντικά την κατανόηση και την ερμηνεία του κριτηρίου. Για το λόγο αυτό πολύ συχνά χρησιμοποιείται η τετραγωνική ρίζα μέσου σφάλματος τετραγώνου (Root Mean Squared Error – RMSE) που εκφράζεται στις ίδιες μονάδες μέτρησης με τις τιμές της χρονοσειράς. Η σχέση που υπάρχει μεταξύ MSE και RMSE θα μπορούσαμε να πούμε ότι είναι αντίστοιχη της σχέσεως που υπάρχει μεταξύ τυπικής απόκλισης και διακύμανσης. β) ο τρόπος με τον οποίο υπολογίζεται το MSE δίνει μεγάλη βαρύτητα στα μεγάλα παρά στα μικρά σφάλματα. Κατά συνέπεια η ύπαρξη προβλέψεων που απέχουν πολύ από τις αντίστοιχες πραγματικές τιμές γίνεται περισσότερο αισθητή με το κριτήριο MSE από ότι με το κριτήριο MAD, γιατί το σφάλμα της πρόβλεψης υψώνεται στο τετράγωνο. γ) Το κριτήριο MSE θεωρείται ότι είναι στατιστικά περισσότερο αξιόπιστο από το κριτήριο MAD.

ΜΕΣΟ ΑΠΟΛΥΤΟ ΠΟΣΟΣΤΙΑΙΟ ΣΦΑΛΜΑ (MEAN ABSOLUTE PERCENTAGE ERROR – MAPE)

Ένα άλλο κριτήριο αξιολόγησης των μεθόδων προβλέψεων είναι το μέσο απόλυτο ποσοστιαίο σφάλμα, το οποίο εξετάζει τη συμπεριφορά της απόλυτης τιμή του σφάλματος της πρόβλεψης σε σχέση με την πραγματική τιμή της χρονοσειράς. Το μέσο απόλυτο ποσοστιαίο σφάλμα ορίζεται ως το άθροισμα των απόλυτων τιμών των σφαλμάτων των χρονοσειρών προς τις αντίστοιχες πραγματικές τιμές της χρονοσειράς, διαιρούμενο με τον αριθμό των χρονικών περιόδων n στις οποίες έγιναν προβλέψεις. Δηλαδή :

$$\mathrm{M} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|,$$

όπου At η πραγματική τιμή και Ft η εκτιμώμενη τιμή.

Σημείωση: μετράει το πόσο έξω πέφτουν οι προβλέψεις ως ποσοστά της πραγματικής τιμής της μεταβλητής

MAPE = 1 / n * [Σt(IetI / Yt)], Όπου t =1,2,…,n

Το συγκεκριμένο κριτήριο δείχνει πόσο μεγάλο είναι το σφάλμα της πρόβλεψης σε σχέση με τις πραγματικές τιμές της χρονοσειράς. Όσο μικρότερη είναι η τιμή που λαμβάνει τόσο πιο καλή θεωρείται η μέθοδος πρόβλεψης. Επειδή το κριτήριο MAPE είναι απαλλαγμένο από μονάδες μέτρησης, χρησιμοποιείται για να συγκρίνουμε την ακρίβεια μιας ή περισσοτέρων μεθόδων πρόβλεψης σε δύο ή περισσότερες χρονοσειρές.

ΜΕΣΟ ΠΟΣΟΣΤΙΑΙΟ ΣΦΑΛΜΑ (MEAN PERCENTAGE ERROR –MPE)

Όταν θέλουμε να προσδιορίσουμε κατά πόσο οι προβλεπόμενες τιμές είναι συστηματικά μεγαλύτερες ή μικρότερες από τις αντίστοιχες πραγματικές τιμές της χρονοσειράς, δηλαδή κατά πόσο η μέθοδος πρόβλεψης που εφαρμόζουμε είναι μεροληπτική ή όχι, μπορούμε να εφαρμόσουμε το μέσο ποσοστιαίο σφάλμα. Ο τρόπος υπολογισμού του κριτηρίου αυτού είναι ακριβώς ίδιος με τον τρόπο υπολογισμού του μέσου απόλυτου ποσοστιαίου σφάλματος, με τη μόνη διαφορά ότι εδώ δεν χρησιμοποιούμε τις απόλυτες αλλά τις πραγματικές τιμές του σφάλματος της πρόβλεψης. Δηλαδή :

MPE = 1 / n * [Σt(et / Yt)], Όπου t =1,2,…,n

Σημείωση: μέσο σφάλμα επί τοις 100 για N περιόδους.

Όταν οι τιμές που λαμβάνει το κριτήριο αυτό είναι κοντά στο μηδέν, τότε η μέθοδος πρόβλεψης θεωρείται αμερόληπτη. Αντιθέτως, μεγάλες τιμές του κριτηρίου καταδεικνύουν μεγάλη μεροληψία. Αρνητική τιμή του MPE σημαίνει ότι η μέθοδος πρόβλεψης παρέχει υπερεκτιμημένες προβλέψεις σε σχέση με τις πραγματικές τιμές, ενώ θετική τιμή του MPE σημαίνει ότι οι τιμές της χρονοσειράς είναι υποεκτιμημένες.

ΡΙΖΑ ΜΕΣΟΥ ΤΕΤΡΑΓΩΝΙΚΟΥ ΣΦΑΛΜΑΤΟΣ (Root Mean squared error, RMSE) δίνεται από τον τύπο:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N} E_t^2}{N}}$$

Συμπερασματικά, για να προσδιορίσουμε την αξιοπιστία μιας συγκεκριμένης μεθόδου πρόβλεψης, θα πρέπει να μελετήσουμε τη διαχρονική συμπεριφορά των τιμών των σφαλμάτων της πρόβλεψης. Αυτό γίνεται με την εφαρμογή διάφορων κριτηρίων, σύμφωνα με τα οποία αξιολογούμε τη χρησιμοποιούμενη μέθοδο πρόβλεψης. Κάθε ένα από τα κριτήρια αυτά ορίζεται από μία συγκεκριμένη συναρτησιακή σχέση των σφαλμάτων της πρόβλεψης και μπορεί να χρησιμοποιηθεί όχι μόνο για την αξιολόγηση μιας μεθόδου πρόβλεψης αλλά και για την επιλογή της " καλύτερης " μεταξύ δύο ή περισσοτέρων εναλλακτικών μεθόδων προβλέψεων.

Παρατήρηση: Τους μαθηματικούς τύπους στους οποίους αναφερθήκαμε παραπάνω τους βρήκαμε μέσω του internet από την ιστοσελίδα του Wikipedia: http://en.wikipedia.org/wiki/Moving_average

Σύγκριση Μεθόδων Πρόβλεψης Τιμών Μετοχών

Moving – Cumulative average

Πλεονεκτήματα

- Εύκολα κατανοητή
- Υπολογίζεται εύκολα
- Παρέχει σταθερές προβλέψεις

Μειονεκτήματα

- Προϋποθέτει την αποθήκευση πολλών από τα προηγούμενα σημεία δεδομένων: τουλάχιστον οι N περίοδοι χρησιμοποιούνται για τον υπολογισμό του κινούμενου μέσου όρου
- Υστερεί σε σχέση με την τάση
- Αγνοεί πολύπλοκες σχέσεις των δεδομένων

Weighted Moving Averages

Αυτή η μέθοδος εξετάζει δεδομένα του παρελθόντος και προσπαθεί να αποδίδουν λογικά σημασία σε ορισμένα στοιχεία σε σχέση με άλλα στοιχεία

- Συντελεστές βαρύτητας πρέπει να προσθέσετε σε ένα
- Μπορεί να σταθμίσουμε τα τελευταία υψηλότερα από ό, τι παλαιότερα ή συγκεκριμένα στοιχεία πάνω από τα άλλα
- Σε περίπτωση πρόβλεψη στελέχωσης, θα μπορούσαμε να χρησιμοποιήσουμε τα δεδομένα από τις τελευταίες τέσσερις εβδομάδες, όπου οι Τρίτες είναι που θα προβλέψεις.
- Στάθμιση τις Τρίτες είναι: Τ-1 είναι .25; Τ-2 είναι .20; Τ-3 είναι .15; Τ-4 είναι .10 και Μέσος όρος όλων των άλλων ημερών ζυγίζεται .30

.Exponential Smoothing Method

Ένας τύπος του σταθμισμένου κινητού μέσου όρου που ισχύει για μείωση βαρών ιστορικών δεδομένων

1. . Νέα Πρόγνωση = a (πιο πρόσφατη παρατήρηση) + (1 - α) (τελευταία πρόβλεψη)

ή

2. Νέα Πρόγνωση = τελευταία πρόβλεψη - ένα (το τελευταίο σφάλμα πρόβλεψης)

όπου 0 <α <1 και γενικά είναι μικρό για τη σταθερότητα των προβλέψεων (περίπου 0,1 - 0,2)

Σε σύμβολα:

Ft+1 = a Dt + (1 - a) Ft

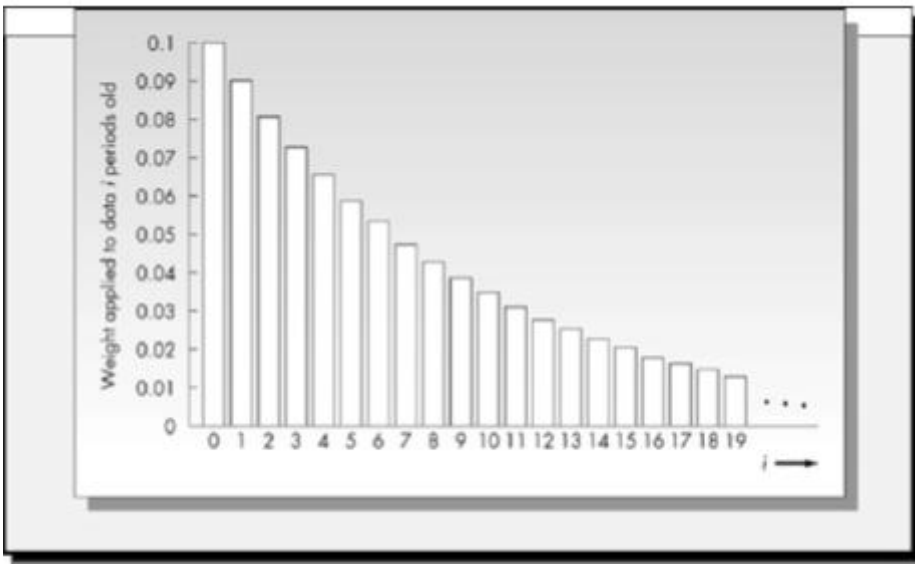
= a Dt + (1 - a) (a Dt-1 + (1 - a) Ft-1)

= a Dt + (1 - a)(a)Dt-1 + (1 - a)2 (a)Dt - 2 + . . .

Ως εκ τούτου, η μέθοδος αυτή εφαρμόζεται ένα σύνολο εκθετικά φθίνουσα βαρών σε ιστορικά δεδομένα. Είναι εύκολο να αποδειχθεί ότι το άθροισμα των βαρών είναι ακριβώς ένα.

{Or : Ft + 1 = Ft - a (Ft - Dt) }

Weights in Exponential Smoothing:



Επίδραση της α τιμής για την Πρόβλεψη

- Μικρές τιμές του α σημαίνει ότι η προβλεπόμενη τιμή θα είναι σταθερές (δείχνουν χαμηλή μεταβλητότητα)
- Χαμηλή α αυξάνει την υστέρηση των προβλέψεων για τα πραγματικά δεδομένα, εάν η τάση είναι παρούσα
- Μεγάλες τιμές α σημαίνει ότι η πρόβλεψη θα παρακολουθούν πιο στενά την πραγματική χρονολογική σειρά

Ας δούμε ένα παράδειγμα:

Jan 23.3 Feb 72.3 Mar 30.3 Apr 15.5

And the January Forecast was: 25

Using α = .15

Forecast for Feb: αDjan + (1- α)Fjan = .15*23.3 + (.85)*25 = 24.745

Forecast for Mar: αDfeb + (1- α)Ffeb = .15*72.3 + (.85)*24.745 = 31.88

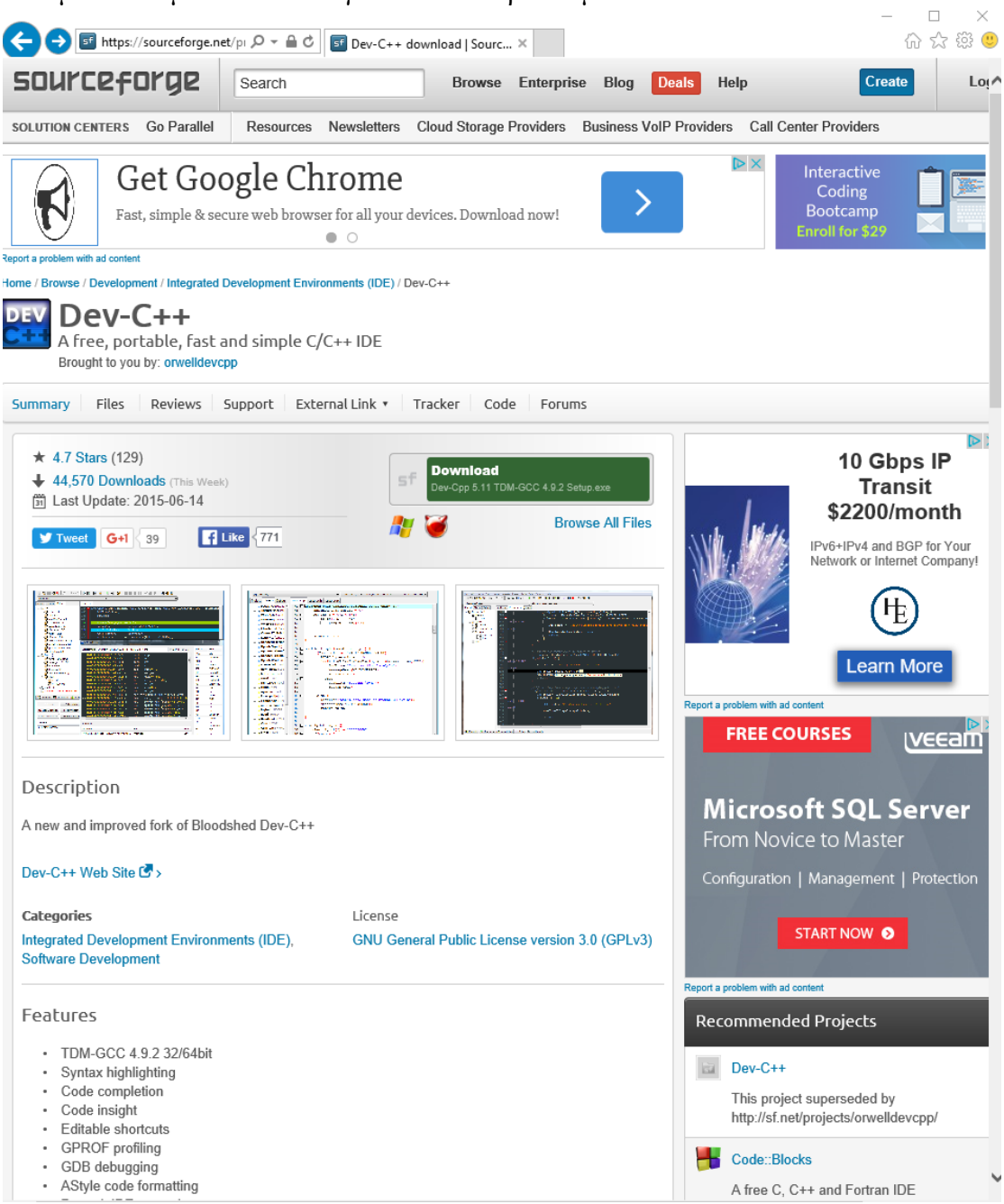
Apr: αDmar + (1- α)Fmar = .15*30.3 + .85*31.88 = 31.64

May: αDapr + (1- α)Fapr = .15*15.5 + .85*31.64 = 29.22

Εγκατάσταση του Dev C++

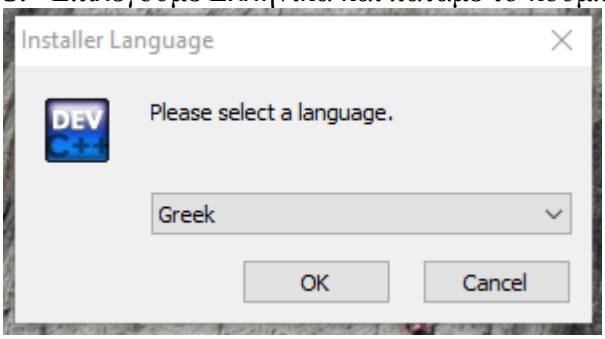
Το Dev ++ είναι ένα χρήσιμο εργαλείο, αφού μας προσφέρει μια πλατφόρμα για να δημιουργούμε προγράμματα και να τα επεξεργαστούμε. Για να το εγκαταστήσουμε λοιπόν, στον υπολογιστή μας ακολουθούμε τα παρακάτω βήματα:

1. Πάμε στο ακόλουθο link https://sourceforge.net/projects/orwelldevcpp/ και πατάμε το κουμπί Download για να το κατεβάσουμε.

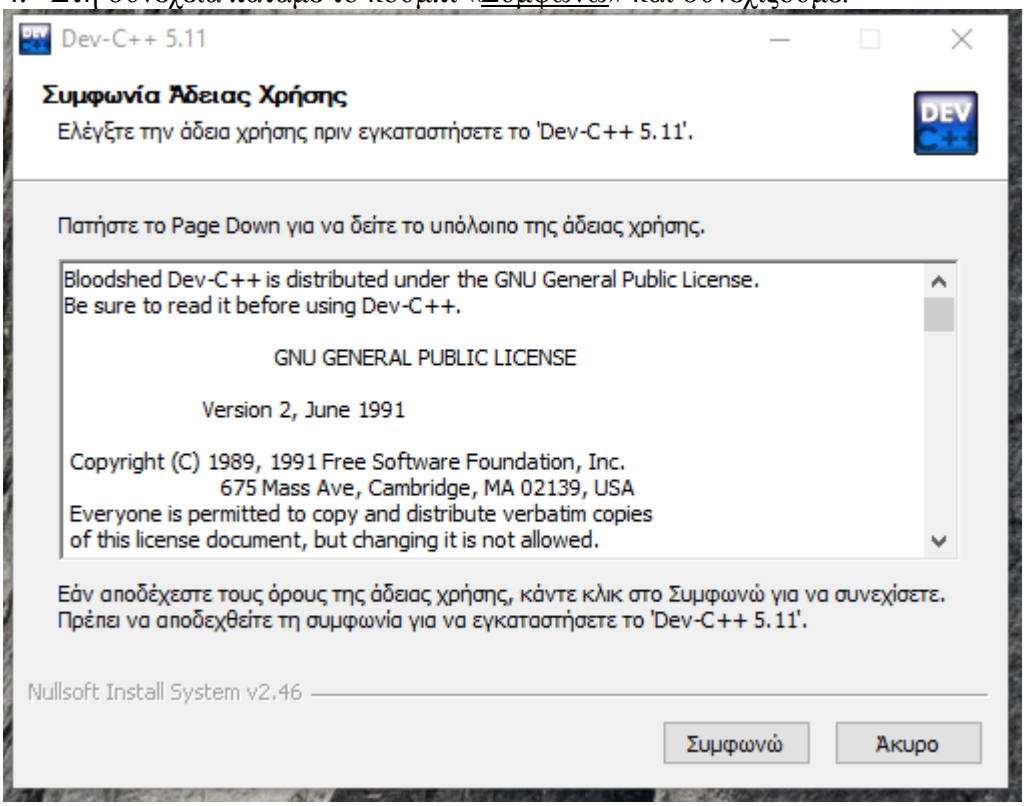


2. Αφού ολοκληρωθεί η λήψη εκτελούμε το αρχείο Dev-Cpp 5.11 TDM-GCC 4.9.2 Setup, στην συνέχεια ακολουθούμε τα βήματα του οδηγού εγκατάστασης.

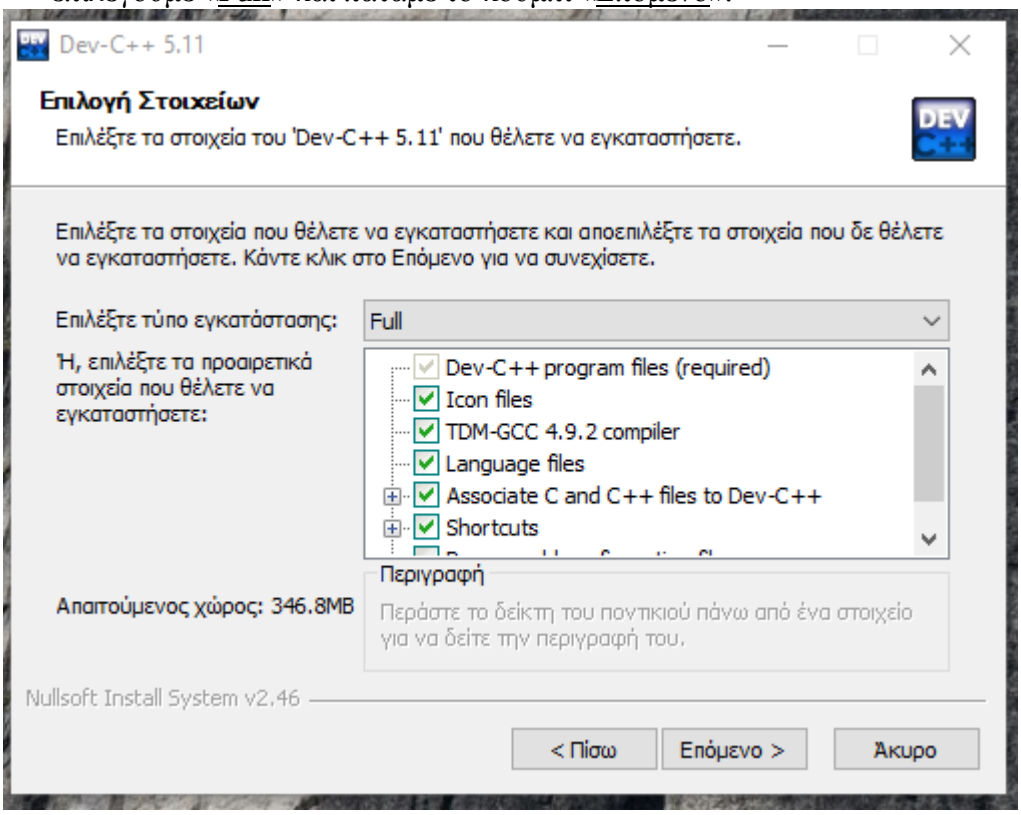
3. Επιλέγουμε Ελληνικά και πατάμε το κουμπί «OK».



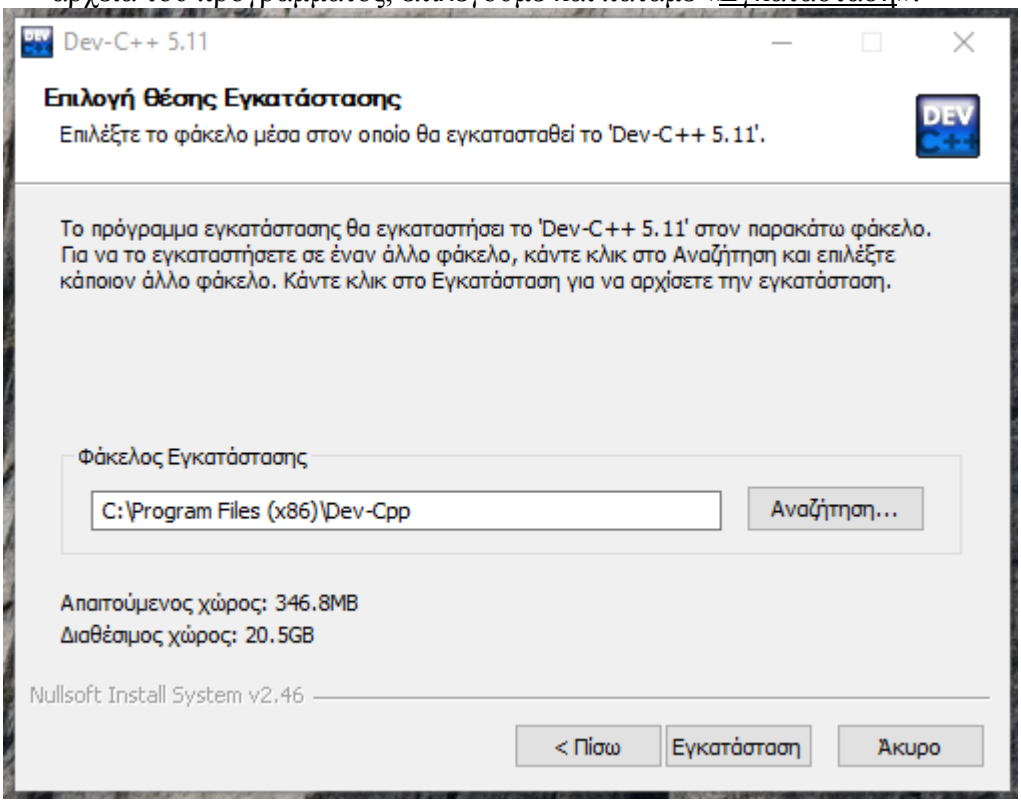
4. Στη συνέχεια πατάμε το κουμπί «Συμφωνώ» και συνεχίζουμε.



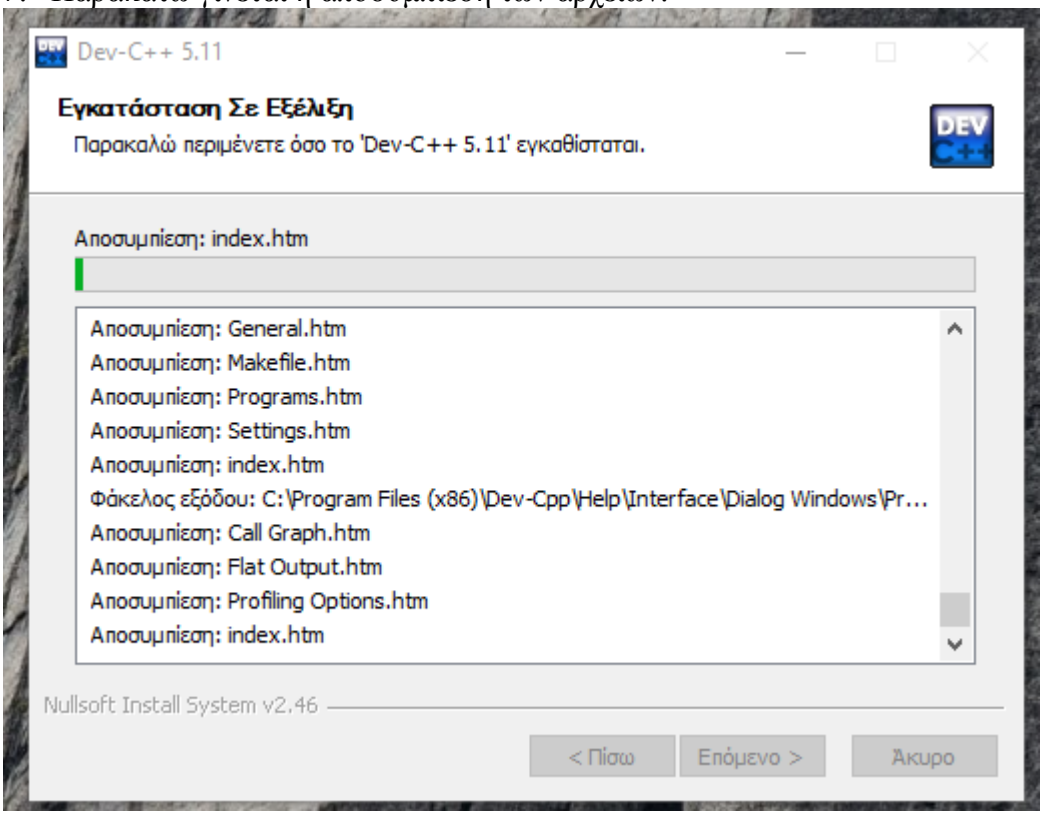
5. Παρακάτω επιλέγουμε τον τύπο της εγκατάστασης που θέλουμε να κάνουμε, επιλέγουμε «Full» και πατάμε το κουμπί «Επόμενο».



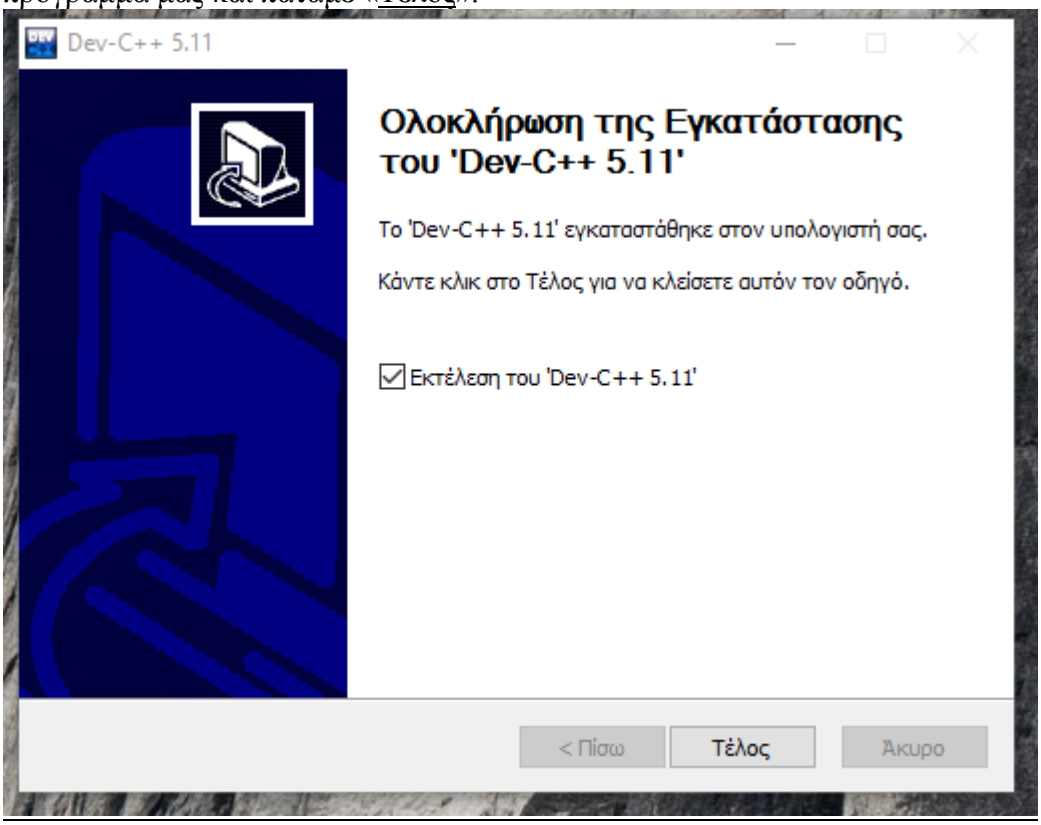
6. Στην συνέχεια επιλέγουμε τον φάκελο που θέλουμε να εγκαταστήσουμε τα αρχεία του προγράμματος, επιλέγουμε και πατάμε «Εγκατάσταση».



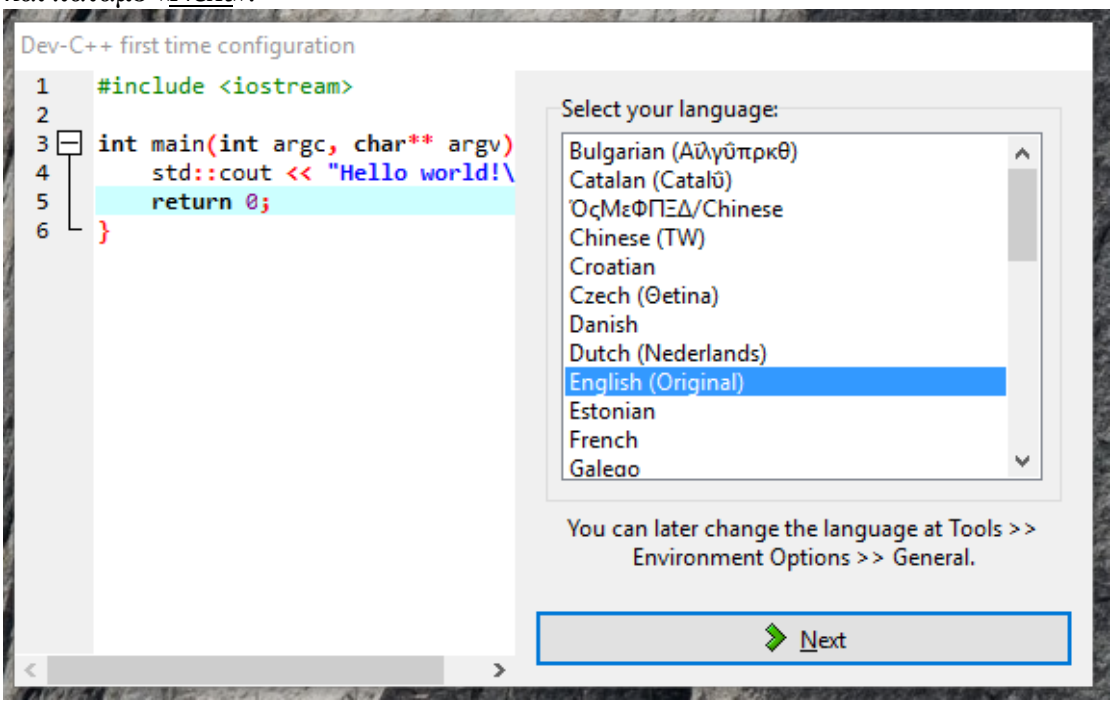
7. Παρακάτω γίνεται η αποσυμπίεση των αρχείων.



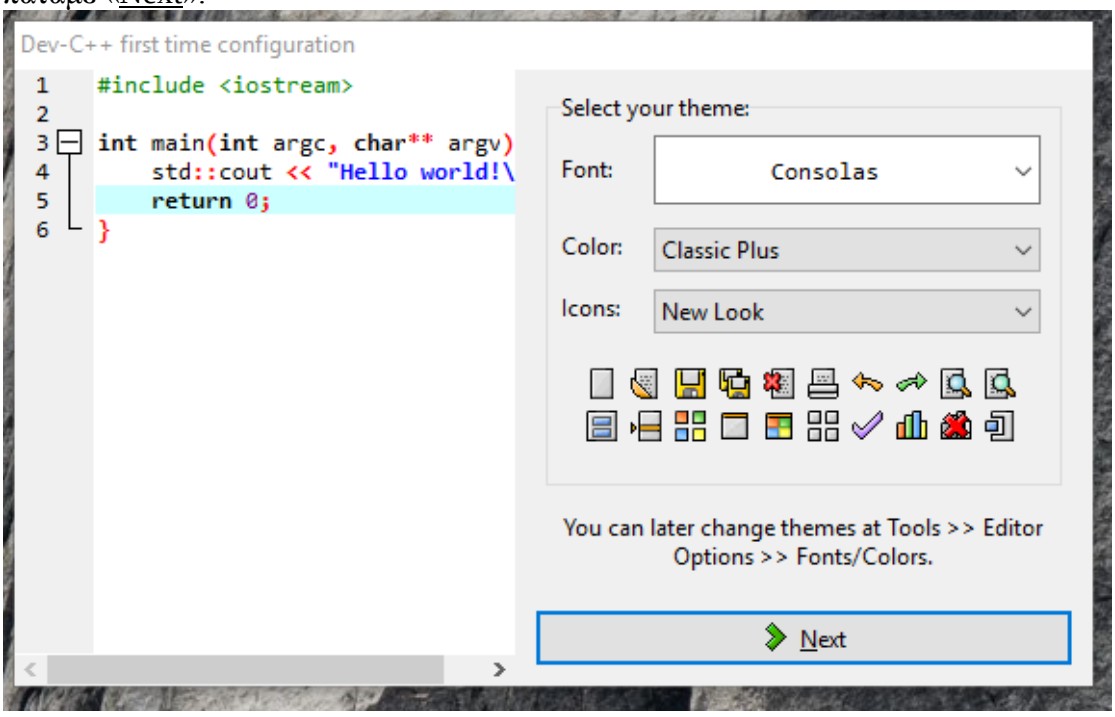
8. Η εγκατάστασή μας ολοκληρώθηκε με επιτυχία. Επιλέγουμε να εκτελεστεί το πρόγραμμά μας και πατάμε «Τέλος».



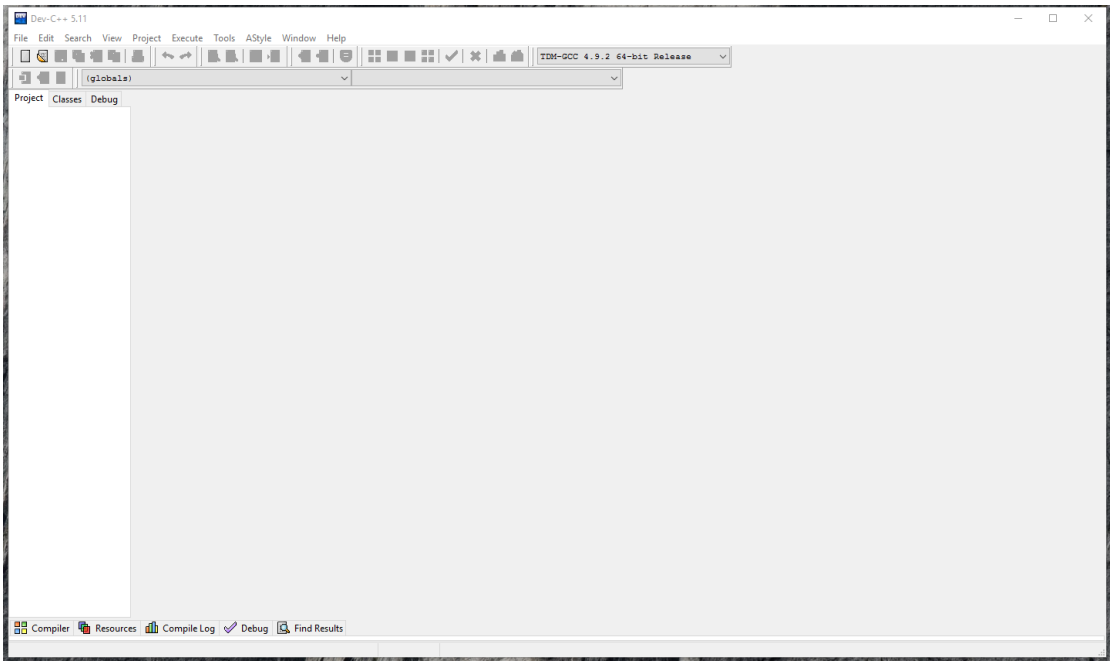
9. Παρακάτω ρυθμίζουμε την γλώσσα που θέλουμε να έχει το πρόγραμμά μας και πατάμε «Next».



10. Παρακάτω ρυθμίζουμε το θέμα που θέλουμε να έχει το πρόγραμμά μας και πατάμε «Next».



11. Τέλος μας εμφανίζει το περιβάλλον εργασίας στο οποίο είμαστε έτοιμοι να δουλέψουμε.


Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Compiler Resources Compile Log Debug Find Results

Υλοποίηση αλγορίθμου K-μέσων(k-means)

Έχουμε ένα αρχείο με 200 τιμές από μετοχές και θέλουμε να τα ομαδοποιήσουμε σε Κ ομάδες. Η ομαδοποίηση γίνεται ως προς σε ποιο κέντρο (x,y) κάθε ομάδας είναι ποιο κοντά το κάθε στοιχείο. Αρχικά περνάμε τα δεδομένα σε μια απλά συνδεδεμένη λίστα (struct metoxi1) Και στην συνέχεια την λίστα ένα πίνακα Ν θέσεων δομών της μορφής struct metoxi. Το πρόγραμμα ζητάει τον αριθμό των ομάδων Κ και αφού το διαβάσει ζητάει για κάθε ομάδα να δώσουμε τα αρχικά κέντρα τους(x,y).
Μετά το πρόγραμμα μπαίνει σε μια while που γίνεται 1 αν υπάρχει αλλαγή στα κέντρα έστω και 1 ομάδα από το προηγούμενο βήμα και οι επαναλήψεις δεν έχουν ξεπεράσει τις 100φορές.
Μετά υπολογίζεται η απόσταση του καθένα από τις 200 μετοχές από το κέντρο της κάθε ομάδας.
Δηλαδή αν έχουμε 6 ομάδες και 200 μετοχές θα υπολογιστούν 6*200 κέντρα.
Στην συνέχεια έχω ένα πίνακα 2 διαστάσεων τον group όπου αποθηκεύω σε ποια ομάδα θα μπει τελικά το στοιχείο (σε αυτή με την μικρότερη απόσταση). Τον πίνακα group τον αρχικοποιώ κάθε φορά που μπαίνω στην while γιατί κάθε φορά αλλάζουν τα κέντρα οπότε και οι ομάδες. Μετά κάνουμε τον έλεγχο και βάζω στον group τα δεδομένα.
Όταν τελειώσει αυτό και πλέον κάθε δεδομένο είναι σε κάποια ομάδα υπολογίζουμε τα νέα κέντρα της κάθε ομάδας.
Το kvalues_x[i]=sum_x[i]/counter μας λέει ότι το νέο κέντρο για την i ομάδα είναι το άθροισμα όλων των X συντεταγμένων των στοιχείων της ομάδας δια το πόσα στοιχεία έχει η ομάδα. Όμοια και το kvalues_y[i]=sum_y[i]/counter.
Πριν γίνει αυτός ο υπολογισμός αποθηκεύουμε το kvalues_x[i] και kvalues_y[i] σε 2 άλλους πίνακες, τους prevkvalues_x[i] και prevkvalues_y[i]. Αυτό γίνεται για να έχουμε τα προηγούμενα κέντρα των ομάδων ώστε να μπορούμε να ελέγχουμε αν έχουμε αλλαγές στις ομάδες από βήμα σε βήμα.
Εδώ εκτυπώνεται κάθε ομάδα ποια στοιχεία ανήκουν καθώς και το νέο κέντρο της κάθε ομάδας.
Τέλος γίνεται ο έλεγχος αν υπάρχει κάποια αλλαγή στις νέες ομάδες από τις προηγούμενες έτσι ώστε όταν πάει ο έλεγχος πάλι στη while να αποφασιστεί αν θα ξαναμπεί ή αν τερματιστεί το πρόγραμμα.

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
#include<math.h>
#define N 200

struct metoxi{
    double simvolo;
    double timi;
    double pososto_metoxis;
    double tziros;

};

struct metoxi1{
    double simvolo;
    double timi;
    double pososto_metoxis;
    double tziros;

    struct metoxi1 *next;
};

struct metoxi1 *insert(struct metoxi1 *,struct metoxi1 *);
struct metoxi1 *make_list_from_file(char *,struct metoxi1 *);
struct metoxi1 *new_node(int,int,int,int);

```

Στο argv[1υπάρχει το όνομα του αρχείου(arxeio2.txt).
Στο argv[0] είναι το εκτελέσιμο της άσκησης(ask2.exe).

```c
main(int argc,char *argv[]){

struct metoxi pin[N],group[N][N],value,val0;
struct metoxi1 *head=NULL;
struct metoxi1 *cur;
int i,k,j,group_number,allagh=1,vhma=1;
double centro_x,centro_y,x=0.0,y=0.0,min;
double kvalues_x[N],kvalues_y[N],prevkvalues_x[N],prevkvalues_y[N];
double apostasi_apo_kentro[N][N];
double sum_x,sum_y,count;
```

Στην περίπτωση μας το argc είναι 2.
Αν βάλουμε λάθος όρισμα τότε το πρόγραμμα κλείνει.

```c
if(argc!=2){
    printf("WRONG number of arguments.\n");
    exit(0);
}
```

Στο argv[1] υπάρχει το όνομα του αρχείου (arxeio2.txt).
Η συνάρτηση δέχεται το arxeio1.txt και την κεφαλή της λίστας.
Επιστρέφει την τελική λίστα.

```c
head=make_list_from_file(argv[1],head);
```

Μεταφέρω την λίστα σε ένα πίνακα Ν θέσεων που περιέχει struct τύπου "struct metoxi".

```
55  cur=head;
56  while(cur!=NULL){
57          pin[i].simvolo=cur->simvolo;
58          pin[i].timi=cur->timi;
59          pin[i].pososto_metoxis=cur->pososto_metoxis;
60          pin[i].tziros=cur->tziros;
61      cur=cur->next;
62      i++;
63  }
```

Δίνω πόσες ομάδες θα έχω για να ομαδοποιήσω τα δεδομένα.

```
78  printf("Dwse ton ari8mo K twn omadwn:");
79  scanf("%d",&k);
```

Δίνω τυχαία κέντρα (X,Y) για κάθε ομάδα.
Τα αποθηκεύω σε 2 πίνακες. Ο πίνακας kvalues_x[i] έχει τα X για κάθε ομάδα
Ενώ ο kvalues_y[i] έχει τα Y. Το i μας δείχνει την ομάδα που οι αντίστοιχες συντεταγμένες
Για παράδειγμα η ομάδα i=1 έχει κέντρο (kvalues_x[i=1],kvalues_y[i=1]).

```
85  for(i=0;i<k;i++){
86      printf("Dwse to kentro X ths %dhs omadas:",i+1);
87      scanf("%lf",&centro_x);
88      kvalues_x[i]=centro_x;
89      printf("Dwse to kentro Y ths %dhs omadas:",i+1);
90      scanf("%lf",&centro_y);
91      kvalues_y[i]=centro_y;
92  }
```

Το «allagh» μας δείχνει αν στο επόμενο βήμα δεν υπάρχει καμιά αλλαγή.
Από το προηγούμενο τότε γίνεται 0 και δεν ξαναμπαίνει στην while, επίσης αν ξεπεράσουμε
τα 100βήματα δεν ξαναμπαίνει στην while.

```
97  while(allagh==1 && vhma <= 100 ){
```

Εδώ βρίσκω για κάθε στοιχείο του αρχείου την απόσταση του από κάθε κέντρο από τις ομάδες που έχω, για παράδειγμα αν έχω 6 ομάδες θα βρω για το κάθε στοιχείο 6 αποστάσεις. Τα αποθηκεύω σε έναν πίνακα 2 διαστάσεων.

```
102 for( i = 0; i < k; i++){
103     for( j = 0; j < N; j++){
104
105
106         x=(kvalues_x[i]-pin[j].pososto_metoxis);      //(x2-x1)
107         y=(kvalues_y[i]-pin[j].tziros);       //(y2-y1)
108         apostasi_apo_kentro[i][j] = sqrt(pow(x,2) + pow(y,2));
109     }
110   }
```

```
125   val0.simvolo=-1.0;
126   val0.timi=-1.0;
127   val0.pososto_metoxis=-1.0;
128   val0.tziros=-1.0;
```

Αρχικοποιώ τον πίνακα που περιέχει τις ομάδες. Βάζω την τιμή -1 που δεν υπάρχει σαν τιμή
στα δεδομένα μας.

```
130   for(i = 0; i <N ; i++){
131       for(j = 0; j < N; j++){
132           group[i][j] = val0;
133       }
134   }
```

Εδώ ελέγχουμε σε ποιο από τις ομάδες έχει μικρότερη απόσταση το κάθε δεδομένο και αποθηκεύω τα δεδομένα στις αντίστοιχες ομάδες.

```
149   }
150
151   printf("\n\t\t\t***Vhma=%d***\n",vhma);
152   for( i = 0; i < k; i++){                        //diatrexw tis omades K
153         sum_x=0.0;                                //sto sum_x pros8etw ola
154         sum_y=0.0;                                //to count(mas dinei pos
155         count=0.0;                                //to neo kentro ths omad
156         printf("Omada%d:\n",i+1);
157
158   for( j = 0; j < N; j++){
159               if(group[i][j].simvolo!=-1){              //an den einai
160                   printf("(%.0lf,%.0lf)\t",group[i][j].pososto_metoxi
161                   sum_x=sum_x + group[i][j].pososto_metoxis;  //pros8
162                   sum_y=sum_y + group[i][j].tziros;
163                   count++;
164               }
165       }
166       prevkvalues_x[i] = kvalues_x[i];              //prin apo8hkeusw
167       prevkvalues_y[i] = kvalues_y[i];              //ton prevkavalues
168                                                     //kapoio kentro h'
169       kvalues_x[i] = sum_x/count;                   //vazw to neo kent
170       kvalues_y[i] = sum_y/count;
171       printf("\n");
172       printf("Neo kentro:(%.2lf,%.2lf).\n",kvalues_x[i],kvalues_y[i]);
173       printf("\n");
174   }
```

Ελέγχει αν έστω και ένα κέντρο είναι διαφορετικό. Κάνουμε αλλαγή το 0 αν δεν μπει στην if σημαίνει ότι τα κέντρα του βήματος αυτού με του προηγούμενου είναι όλα σύμβολα οπότε πρέπει να τερματιστεί το πρόγραμμα.

```
179   allagh=0;
180   for(i=0;i<k;i++){
181                   //prepei kai to X kai to Y tautoxrona na einai to simvoloio me to proigoumeno (X,Y)
182     if( (prevkvalues_x[i] != kvalues_x[i]) || (prevkvalues_y[i] != kvalues_y[i]))
183                   allagh=1;
184
185   }
186   vhma++;
187   }
188
189   system("pause");
190   }
```

Δέχεται τον κόμβο που έχουμε φτιάξει στην συνάρτηση new_node. Η συνάρτηση ελέγχει αν η λίστα είναι κενή(γίνεται μόνο την πρώτη φορά στην πρώτη εισαγωγή).

Αν είναι κενή βάζει την κεφαλή της λίστας να δείχνει στο κόμβο αλλιώς διατρέχει την λίστα, φτάνει στο τελευταίο στοιχείο της και βάζει τον νέο κόμβο, τέλος επιστρέφει την κεφαλή της λίστας.

```
197 struct metoxi1 *insert(struct metoxi1 *head,struct metoxi1 *node){
198
199  struct metoxi1 *current=head;
200
201 if(head==NULL){
202      head=node;
203      return head;
204 }
205
206 else{
207      while(current->next!=NULL)
208          current=current->next;
209
210      current->next=node;
211 }
212
213
214      return head;
215 }
```

Ανοίγω ένα αρχείο και διαβάζω κάθε φορά 1 γραμμή του αρχείου με την strok παίρνω κάθε φορά ότι υπάρχει μέχρι να βρει το \t (tab). Εδώ έχω 4 πράγματα οπότε την καλώ 4 φορές, μετά καλώ την new_node για να φτιάξω τον κόμβο και μετά την insert για να βάλω τον κόμβο στην λίστα.

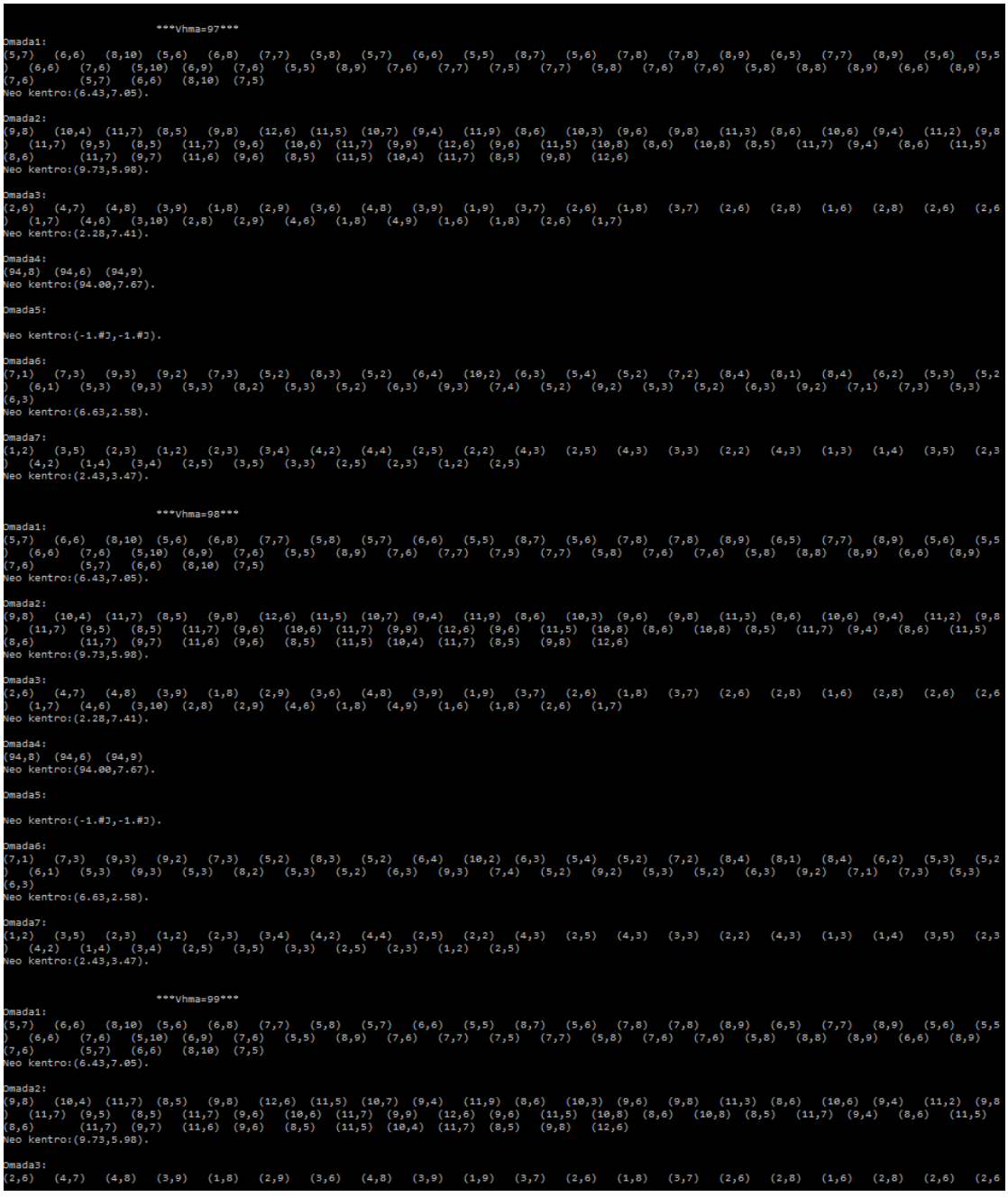
```
222 struct metoxi1 *make_list_from_file(char *filename,struct metoxi1 *head){
223
224 struct metoxi1 *node;
225 char *s=(char *)malloc(100);
226 char *temp;
227
228 int simvolo,timi,megethos,tziros,k=0;
229
230 FILE *fp=fopen(filename,"r");          //anoigw to arxeio me onoma filename (se emas arxeio2.txt)
231                                        //gia na to diavasw mono.. r..
232 if(fp==NULL){                          //an einai adeio tote ektypwnei eror
233     printf("Error at opening File %s\n",filename);
234     exit(0);                           //termatizetai to programa
235 }
236
237
238 while((s=fgets(s,100,fp))!=NULL){   //apo8hkevw thn grammh pou pernw ka8e fora sto s me megistw mhkos
239                                     //dinw megalo mhkos gia na diavasei olh thn grammh
240     k++;                            //metraw poses grammes exei to arxeio
241     simvolo=k;                          //to simvolo dixnei thn grammh tou arxeiou
242
243
244     if((temp=strtok(s,"\t"))!=NULL)  //prwth fora kaleitai me to s kai to /t.meta panta me NULL kai /
245         timi=atoi(temp);
246     else{
247         printf("Error1 at reading file.\n");  //ton elegxo ton kanw gia na diasfalisw oti to arxeio e
248         exit(0);                              //dedomena...px den yparxei grammh me 5 ari8mous
249     }
250
251
252     if((temp=strtok(NULL,"\t"))!=NULL)
253         megethos=atoi(temp);
254
255     else{
256         printf("Error2 at reading file.\n");
257         exit(0);
258     }
259
260     if((temp=strtok(NULL,"\t"))!=NULL)
261         tziros=atoi(temp);
262     else{
263         printf("Error3 at reading file.\n");
264         exit(0);
265     }
266
267
268     node=new_node(simvolo,timi,megethos,tziros);  //ftiaxnw ena neo  komvo-  paketo me ta 4 dedomena
269
270      head=insert(head,node);                    //eisagw to komvo - paketo sthn apla syndedemenh list
271
272 }
273 free(s);                                       //apeleu8erwnw thn mnhmh gia to s
274 fclose(fp);                                    //kleinw to arxeio
```

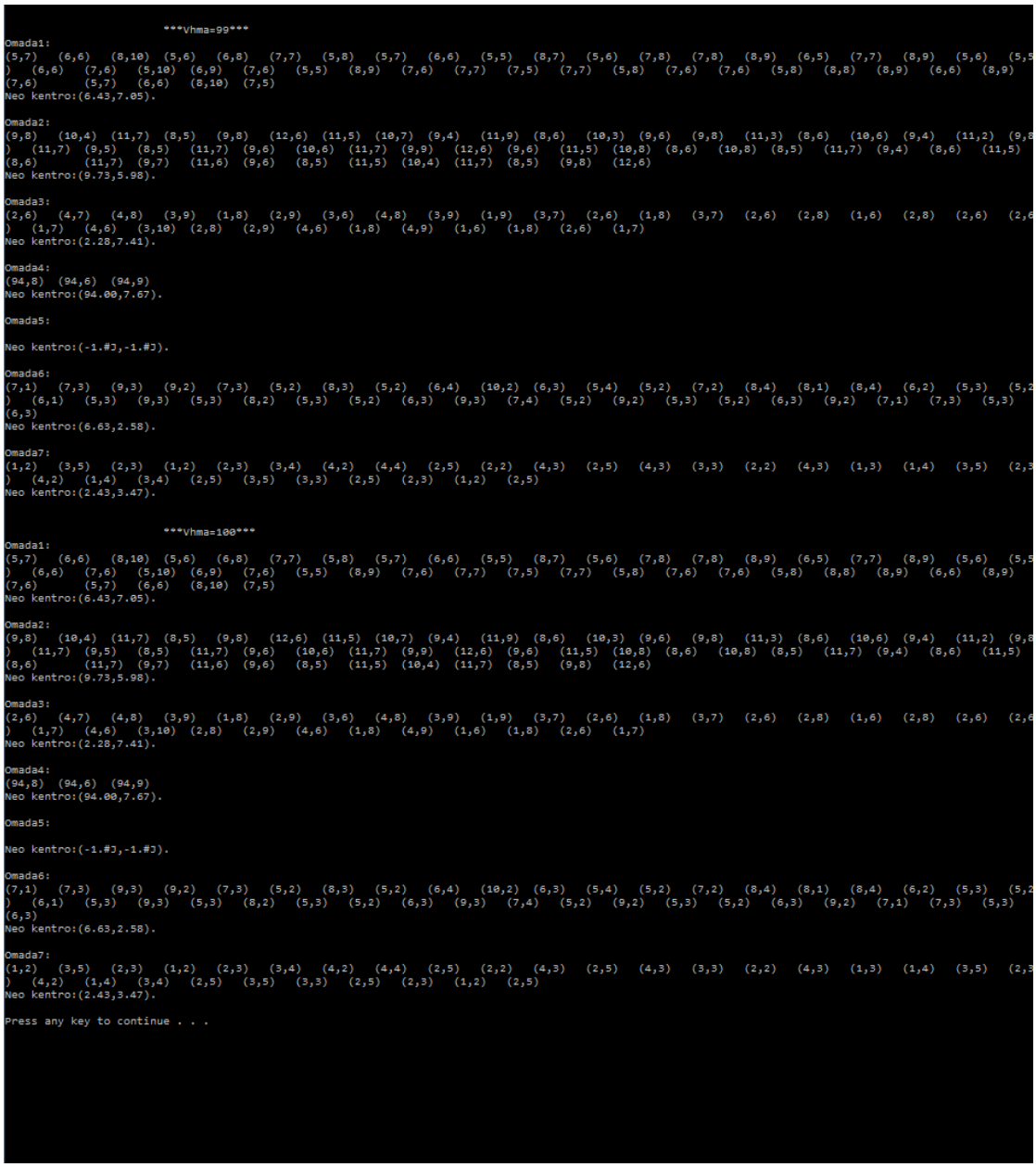
Παίρνω τα 4 στοιχεία και φτιάχνω ένα κόμβο, τον δείχτη τον κάνω NULL και επιστρέφω τον κόμβο.

```
281 struct metoxi1 *new_node(int simvolo,int timi,int megethos,int tziros){
282
283     struct metoxi1 * tmp=(struct metoxi1 *)malloc(sizeof(struct metoxi1));
284
285     tmp->simvolo=simvolo;
286     tmp->timi=timi;
287     tmp->pososto_metoxis=megethos;
288     tmp->tziros=tziros;
289
290     tmp->next=NULL;
291
292 return tmp;
293 }
294
```

Παρακάτω βλέπουμε κάποια παραδείγματα και τα αποτελέσματα που μας δίνει ο αλγόριθμος K-μέσων. Επίσης φαίνονται κάποια από τα βήματα που «ακολουθεί» ο αλγόριθμός μας. Στο παράδειγμα χρησιμοποιήσαμε ένα αρχείο(.txt) με τις μετοχές μας.

```
Omada5:

Neo kentro:(-1.#J,-1.#J).

Omada6:
(7,1)  (7,3)  (9,3)  (9,2)  (7,3)  (5,2)  (8,3)  (5,2)  (6,4)  (10,2)  (6,3)  (5,4)  (5,2)  (7,2)  (8,4)  (8,1)  (8,4)  (6,2)  (5,3)  (5,2
)  (6,1)  (5,3)  (9,3)  (5,3)  (8,2)  (5,3)  (5,2)  (6,3)  (9,3)  (7,4)  (5,2)  (9,2)  (5,3)  (5,2)  (6,3)  (9,2)  (7,1)  (7,3)  (5,3)
(6,3)
Neo kentro:(6.63,2.58).

Omada7:
(1,2)  (3,5)  (2,3)  (1,2)  (2,3)  (3,4)  (4,2)  (4,4)  (2,5)  (2,2)  (4,3)  (2,5)  (4,3)  (3,3)  (2,2)  (4,3)  (1,3)  (1,4)  (3,5)  (2,3
)  (4,2)  (1,4)  (3,4)  (2,5)  (3,5)  (3,3)  (2,5)  (2,3)  (1,2)  (2,5)
Neo kentro:(2.43,3.47).


                    ***Vhma=95***
Omada1:
(5,7)  (6,6)  (8,10)  (5,6)  (6,8)  (7,7)  (5,8)  (5,7)  (6,6)  (5,5)  (8,7)  (5,6)  (7,8)  (7,8)  (8,9)  (6,5)  (7,7)  (8,9)  (5,6)  (5,5
)  (6,6)  (7,6)  (5,10)  (6,9)  (7,6)  (5,5)  (8,9)  (7,6)  (7,7)  (7,5)  (7,7)  (5,8)  (7,6)  (7,6)  (5,8)  (8,8)  (8,9)  (6,6)  (8,9)
(7,6)  (5,7)  (6,6)  (8,10)  (7,5)
Neo kentro:(6.43,7.05).

Omada2:
(9,8)  (10,4)  (11,7)  (8,5)  (9,8)  (12,6)  (11,5)  (10,7)  (9,4)  (11,9)  (8,6)  (10,3)  (9,6)  (9,8)  (11,3)  (8,6)  (10,6)  (9,4)  (11,2)  (9,8
)  (11,7)  (9,5)  (8,5)  (11,7)  (9,6)  (10,6)  (11,7)  (9,9)  (12,6)  (9,6)  (11,5)  (10,8)  (8,6)  (10,8)  (8,5)  (11,7)  (9,4)  (8,6)  (11,5)
(8,6)  (11,7)  (9,7)  (11,6)  (9,6)  (8,5)  (11,5)  (10,4)  (11,7)  (8,5)  (9,8)  (12,6)
Neo kentro:(9.73,5.98).

Omada3:
(2,6)  (4,7)  (4,8)  (3,9)  (1,8)  (2,9)  (3,6)  (4,8)  (3,9)  (1,9)  (3,7)  (2,6)  (1,8)  (3,7)  (2,6)  (2,8)  (1,6)  (2,8)  (2,6)  (2,6
)  (1,7)  (4,6)  (3,10)  (2,8)  (2,9)  (4,6)  (1,8)  (4,9)  (1,6)  (1,8)  (2,6)  (1,7)
Neo kentro:(2.28,7.41).

Omada4:
(94,8)  (94,6)  (94,9)
Neo kentro:(94.00,7.67).

Omada5:

Neo kentro:(-1.#J,-1.#J).

Omada6:
(7,1)  (7,3)  (9,3)  (9,2)  (7,3)  (5,2)  (8,3)  (5,2)  (6,4)  (10,2)  (6,3)  (5,4)  (5,2)  (7,2)  (8,4)  (8,1)  (8,4)  (6,2)  (5,3)  (5,2
)  (6,1)  (5,3)  (9,3)  (5,3)  (8,2)  (5,3)  (5,2)  (6,3)  (9,3)  (7,4)  (5,2)  (9,2)  (5,3)  (5,2)  (6,3)  (9,2)  (7,1)  (7,3)  (5,3)
(6,3)
Neo kentro:(6.63,2.58).

Omada7:
(1,2)  (3,5)  (2,3)  (1,2)  (2,3)  (3,4)  (4,2)  (4,4)  (2,5)  (2,2)  (4,3)  (2,5)  (4,3)  (3,3)  (2,2)  (4,3)  (1,3)  (1,4)  (3,5)  (2,3
)  (4,2)  (1,4)  (3,4)  (2,5)  (3,5)  (3,3)  (2,5)  (2,3)  (1,2)  (2,5)
Neo kentro:(2.43,3.47).


                    ***Vhma=96***
Omada1:
(5,7)  (6,6)  (8,10)  (5,6)  (6,8)  (7,7)  (5,8)  (5,7)  (6,6)  (5,5)  (8,7)  (5,6)  (7,8)  (7,8)  (8,9)  (6,5)  (7,7)  (8,9)  (5,6)  (5,5
)  (6,6)  (7,6)  (5,10)  (6,9)  (7,6)  (5,5)  (8,9)  (7,6)  (7,7)  (7,5)  (7,7)  (5,8)  (7,6)  (7,6)  (5,8)  (8,8)  (8,9)  (6,6)  (8,9)
(7,6)  (5,7)  (6,6)  (8,10)  (7,5)
Neo kentro:(6.43,7.05).

Omada2:
(9,8)  (10,4)  (11,7)  (8,5)  (9,8)  (12,6)  (11,5)  (10,7)  (9,4)  (11,9)  (8,6)  (10,3)  (9,6)  (9,8)  (11,3)  (8,6)  (10,6)  (9,4)  (11,2)  (9,8
)  (11,7)  (9,5)  (8,5)  (11,7)  (9,6)  (10,6)  (11,7)  (9,9)  (12,6)  (9,6)  (11,5)  (10,8)  (8,6)  (10,8)  (8,5)  (11,7)  (9,4)  (8,6)  (11,5)
(8,6)  (11,7)  (9,7)  (11,6)  (9,6)  (8,5)  (11,5)  (10,4)  (11,7)  (8,5)  (9,8)  (12,6)
Neo kentro:(9.73,5.98).

Omada3:
(2,6)  (4,7)  (4,8)  (3,9)  (1,8)  (2,9)  (3,6)  (4,8)  (3,9)  (1,9)  (3,7)  (2,6)  (1,8)  (3,7)  (2,6)  (2,8)  (1,6)  (2,8)  (2,6)  (2,6
)  (1,7)  (4,6)  (3,10)  (2,8)  (2,9)  (4,6)  (1,8)  (4,9)  (1,6)  (1,8)  (2,6)  (1,7)
Neo kentro:(2.28,7.41).

Omada4:
(94,8)  (94,6)  (94,9)
Neo kentro:(94.00,7.67).

Omada5:

Neo kentro:(-1.#J,-1.#J).

Omada6:
(7,1)  (7,3)  (9,3)  (9,2)  (7,3)  (5,2)  (8,3)  (5,2)  (6,4)  (10,2)  (6,3)  (5,4)  (5,2)  (7,2)  (8,4)  (8,1)  (8,4)  (6,2)  (5,3)  (5,2
)  (6,1)  (5,3)  (9,3)  (5,3)  (8,2)  (5,3)  (5,2)  (6,3)  (9,3)  (7,4)  (5,2)  (9,2)  (5,3)  (5,2)  (6,3)  (9,2)  (7,1)  (7,3)  (5,3)
(6,3)
Neo kentro:(6.63,2.58).

Omada7:
(1,2)  (3,5)  (2,3)  (1,2)  (2,3)  (3,4)  (4,2)  (4,4)  (2,5)  (2,2)  (4,3)  (2,5)  (4,3)  (3,3)  (2,2)  (4,3)  (1,3)  (1,4)  (3,5)  (2,3
)  (4,2)  (1,4)  (3,4)  (2,5)  (3,5)  (3,3)  (2,5)  (2,3)  (1,2)  (2,5)
Neo kentro:(2.43,3.47).


                    ***Vhma=97***
```


Vhma=97
Omada1:
(5,7) (6,6) (8,10) (5,6) (6,8) (7,7) (5,8) (5,7) (6,6) (5,5) (8,7) (5,6) (7,8) (7,8) (8,9) (6,5) (7,7) (8,9) (5,6) (5,5
) (6,6) (7,6) (5,10) (6,9) (7,6) (5,5) (8,9) (7,6) (7,7) (7,5) (7,7) (5,8) (7,6) (7,6) (5,8) (8,8) (8,9) (6,6) (8,9)
(7,6) (5,7) (6,6) (8,10) (7,5)
Neo kentro:(6.43,7.05).
Omada2:
(9,8) (10,4) (11,7) (8,5) (9,8) (12,6) (11,5) (10,7) (9,4) (11,9) (8,6) (10,3) (9,6) (9,8) (11,3) (8,6) (10,6) (9,4) (11,2) (9,8
) (11,7) (9,5) (8,5) (11,7) (9,6) (10,6) (11,7) (9,9) (12,6) (9,6) (11,5) (10,8) (8,6) (10,8) (8,5) (11,7) (9,4) (8,6) (11,5)
(8,6) (11,7) (9,7) (11,6) (9,6) (8,5) (11,5) (10,4) (11,7) (8,5) (9,8) (12,6)
Neo kentro:(9.73,5.98).
Omada3:
(2,6) (4,7) (4,8) (3,9) (1,8) (2,9) (3,6) (4,8) (3,9) (1,9) (3,7) (2,6) (1,8) (3,7) (2,6) (2,8) (1,6) (2,8) (2,6) (2,6
) (1,7) (4,6) (3,10) (2,8) (2,9) (4,6) (1,8) (4,9) (1,6) (1,8) (2,6) (1,7)
Neo kentro:(2.28,7.41).
Omada4:
(94,8) (94,6) (94,9)
Neo kentro:(94.00,7.67).
Omada5:
Neo kentro:(-1.#J,-1.#J).
Omada6:
(7,1) (7,3) (9,3) (9,2) (7,3) (5,2) (8,3) (5,2) (6,4) (10,2) (6,3) (5,4) (5,2) (7,2) (8,4) (8,1) (8,4) (6,2) (5,3) (5,2
) (6,1) (5,3) (9,3) (5,3) (8,2) (5,3) (5,2) (6,3) (9,3) (7,4) (5,2) (9,2) (5,3) (5,2) (6,3) (9,2) (7,1) (7,3) (5,3)
(6,3)
Neo kentro:(6.63,2.58).
Omada7:
(1,2) (3,5) (2,3) (1,2) (2,3) (3,4) (4,2) (4,4) (2,5) (2,2) (4,3) (2,5) (4,3) (3,3) (2,2) (4,3) (1,3) (1,4) (3,5) (2,3
) (4,2) (1,4) (3,4) (2,5) (3,5) (3,3) (2,5) (2,3) (1,2) (2,5)
Neo kentro:(2.43,3.47).
Vhma=98
Omada1:
(5,7) (6,6) (8,10) (5,6) (6,8) (7,7) (5,8) (5,7) (6,6) (5,5) (8,7) (5,6) (7,8) (7,8) (8,9) (6,5) (7,7) (8,9) (5,6) (5,5
) (6,6) (7,6) (5,10) (6,9) (7,6) (5,5) (8,9) (7,6) (7,7) (7,5) (7,7) (5,8) (7,6) (7,6) (5,8) (8,8) (8,9) (6,6) (8,9)
(7,6) (5,7) (6,6) (8,10) (7,5)
Neo kentro:(6.43,7.05).
Omada2:
(9,8) (10,4) (11,7) (8,5) (9,8) (12,6) (11,5) (10,7) (9,4) (11,9) (8,6) (10,3) (9,6) (9,8) (11,3) (8,6) (10,6) (9,4) (11,2) (9,8
) (11,7) (9,5) (8,5) (11,7) (9,6) (10,6) (11,7) (9,9) (12,6) (9,6) (11,5) (10,8) (8,6) (10,8) (8,5) (11,7) (9,4) (8,6) (11,5)
(8,6) (11,7) (9,7) (11,6) (9,6) (8,5) (11,5) (10,4) (11,7) (8,5) (9,8) (12,6)
Neo kentro:(9.73,5.98).
Omada3:
(2,6) (4,7) (4,8) (3,9) (1,8) (2,9) (3,6) (4,8) (3,9) (1,9) (3,7) (2,6) (1,8) (3,7) (2,6) (2,8) (1,6) (2,8) (2,6) (2,6
) (1,7) (4,6) (3,10) (2,8) (2,9) (4,6) (1,8) (4,9) (1,6) (1,8) (2,6) (1,7)
Neo kentro:(2.28,7.41).
Omada4:
(94,8) (94,6) (94,9)
Neo kentro:(94.00,7.67).
Omada5:
Neo kentro:(-1.#J,-1.#J).
Omada6:
(7,1) (7,3) (9,3) (9,2) (7,3) (5,2) (8,3) (5,2) (6,4) (10,2) (6,3) (5,4) (5,2) (7,2) (8,4) (8,1) (8,4) (6,2) (5,3) (5,2
) (6,1) (5,3) (9,3) (5,3) (8,2) (5,3) (5,2) (6,3) (9,3) (7,4) (5,2) (9,2) (5,3) (5,2) (6,3) (9,2) (7,1) (7,3) (5,3)
(6,3)
Neo kentro:(6.63,2.58).
Omada7:
(1,2) (3,5) (2,3) (1,2) (2,3) (3,4) (4,2) (4,4) (2,5) (2,2) (4,3) (2,5) (4,3) (3,3) (2,2) (4,3) (1,3) (1,4) (3,5) (2,3
) (4,2) (1,4) (3,4) (2,5) (3,5) (3,3) (2,5) (2,3) (1,2) (2,5)
Neo kentro:(2.43,3.47).
Vhma=99
Omada1:
(5,7) (6,6) (8,10) (5,6) (6,8) (7,7) (5,8) (5,7) (6,6) (5,5) (8,7) (5,6) (7,8) (7,8) (8,9) (6,5) (7,7) (8,9) (5,6) (5,5
) (6,6) (7,6) (5,10) (6,9) (7,6) (5,5) (8,9) (7,6) (7,7) (7,5) (7,7) (5,8) (7,6) (7,6) (5,8) (8,8) (8,9) (6,6) (8,9)
(7,6) (5,7) (6,6) (8,10) (7,5)
Neo kentro:(6.43,7.05).
Omada2:
(9,8) (10,4) (11,7) (8,5) (9,8) (12,6) (11,5) (10,7) (9,4) (11,9) (8,6) (10,3) (9,6) (9,8) (11,3) (8,6) (10,6) (9,4) (11,2) (9,8
) (11,7) (9,5) (8,5) (11,7) (9,6) (10,6) (11,7) (9,9) (12,6) (9,6) (11,5) (10,8) (8,6) (10,8) (8,5) (11,7) (9,4) (8,6) (11,5)
(8,6) (11,7) (9,7) (11,6) (9,6) (8,5) (11,5) (10,4) (11,7) (8,5) (9,8) (12,6)
Neo kentro:(9.73,5.98).
Omada3:
(2,6) (4,7) (4,8) (3,9) (1,8) (2,9) (3,6) (4,8) (3,9) (1,9) (3,7) (2,6) (1,8) (3,7) (2,6) (2,8) (1,6) (2,8) (2,6) (2,6


Vhma=99
Omada1:
(5,7) (6,6) (8,10) (5,6) (6,8) (7,7) (5,8) (5,7) (6,6) (5,5) (8,7) (5,6) (7,8) (7,8) (8,9) (6,5) (7,7) (8,9) (5,6) (5,5
) (6,6) (7,6) (5,10) (6,9) (7,6) (5,5) (8,9) (7,6) (7,7) (7,5) (7,7) (5,8) (7,6) (7,6) (5,8) (8,8) (8,9) (6,6) (8,9)
(7,6) (5,7) (6,6) (8,10) (7,5)
Neo kentro:(6.43,7.05).
Omada2:
(9,8) (10,4) (11,7) (8,5) (9,8) (12,6) (11,5) (10,7) (9,4) (11,9) (8,6) (10,3) (9,6) (9,8) (11,3) (8,6) (10,6) (9,4) (11,2) (9,8
) (11,7) (9,5) (8,5) (11,7) (9,6) (10,6) (11,7) (9,9) (12,6) (9,6) (11,5) (10,8) (8,6) (10,8) (8,5) (11,7) (9,4) (8,6) (11,5)
(8,6) (11,7) (9,7) (11,6) (9,6) (8,5) (11,5) (10,4) (11,7) (8,5) (9,8) (12,6)
Neo kentro:(9.73,5.98).
Omada3:
(2,6) (4,7) (4,8) (3,9) (1,8) (2,9) (3,6) (4,8) (3,9) (1,9) (3,7) (2,6) (1,8) (3,7) (2,6) (2,8) (1,6) (2,8) (2,6) (2,6
) (1,7) (4,6) (3,10) (2,8) (2,9) (4,6) (1,8) (4,9) (1,6) (1,8) (2,6) (1,7)
Neo kentro:(2.28,7.41).
Omada4:
(94,8) (94,6) (94,9)
Neo kentro:(94.00,7.67).
Omada5:
Neo kentro:(-1.#J,-1.#J).
Omada6:
(7,1) (7,3) (9,3) (9,2) (7,3) (5,2) (8,3) (5,2) (6,4) (10,2) (6,3) (5,4) (5,2) (7,2) (8,4) (8,1) (8,4) (6,2) (5,3) (5,2
) (6,1) (5,3) (9,3) (5,3) (8,2) (5,3) (5,2) (6,3) (9,3) (7,4) (5,2) (9,2) (5,3) (5,2) (6,3) (9,2) (7,1) (7,3) (5,3)
(6,3)
Neo kentro:(6.63,2.58).
Omada7:
(1,2) (3,5) (2,3) (1,2) (2,3) (3,4) (4,2) (4,4) (2,5) (2,2) (4,3) (2,5) (4,3) (3,3) (2,2) (4,3) (1,3) (1,4) (3,5) (2,3
) (4,2) (1,4) (3,4) (2,5) (3,5) (3,3) (2,5) (2,3) (1,2) (2,5)
Neo kentro:(2.43,3.47).
Vhma=100
Omada1:
(5,7) (6,6) (8,10) (5,6) (6,8) (7,7) (5,8) (5,7) (6,6) (5,5) (8,7) (5,6) (7,8) (7,8) (8,9) (6,5) (7,7) (8,9) (5,6) (5,5
) (6,6) (7,6) (5,10) (6,9) (7,6) (5,5) (8,9) (7,6) (7,7) (7,5) (7,7) (5,8) (7,6) (7,6) (5,8) (8,8) (8,9) (6,6) (8,9)
(7,6) (5,7) (6,6) (8,10) (7,5)
Neo kentro:(6.43,7.05).
Omada2:
(9,8) (10,4) (11,7) (8,5) (9,8) (12,6) (11,5) (10,7) (9,4) (11,9) (8,6) (10,3) (9,6) (9,8) (11,3) (8,6) (10,6) (9,4) (11,2) (9,8
) (11,7) (9,5) (8,5) (11,7) (9,6) (10,6) (11,7) (9,9) (12,6) (9,6) (11,5) (10,8) (8,6) (10,8) (8,5) (11,7) (9,4) (8,6) (11,5)
(8,6) (11,7) (9,7) (11,6) (9,6) (8,5) (11,5) (10,4) (11,7) (8,5) (9,8) (12,6)
Neo kentro:(9.73,5.98).
Omada3:
(2,6) (4,7) (4,8) (3,9) (1,8) (2,9) (3,6) (4,8) (3,9) (1,9) (3,7) (2,6) (1,8) (3,7) (2,6) (2,8) (1,6) (2,8) (2,6) (2,6
) (1,7) (4,6) (3,10) (2,8) (2,9) (4,6) (1,8) (4,9) (1,6) (1,8) (2,6) (1,7)
Neo kentro:(2.28,7.41).
Omada4:
(94,8) (94,6) (94,9)
Neo kentro:(94.00,7.67).
Omada5:
Neo kentro:(-1.#J,-1.#J).
Omada6:
(7,1) (7,3) (9,3) (9,2) (7,3) (5,2) (8,3) (5,2) (6,4) (10,2) (6,3) (5,4) (5,2) (7,2) (8,4) (8,1) (8,4) (6,2) (5,3) (5,2
) (6,1) (5,3) (9,3) (5,3) (8,2) (5,3) (5,2) (6,3) (9,3) (7,4) (5,2) (9,2) (5,3) (5,2) (6,3) (9,2) (7,1) (7,3) (5,3)
(6,3)
Neo kentro:(6.63,2.58).
Omada7:
(1,2) (3,5) (2,3) (1,2) (2,3) (3,4) (4,2) (4,4) (2,5) (2,2) (4,3) (2,5) (4,3) (3,3) (2,2) (4,3) (1,3) (1,4) (3,5) (2,3
) (4,2) (1,4) (3,4) (2,5) (3,5) (3,3) (2,5) (2,3) (1,2) (2,5)
Neo kentro:(2.43,3.47).
Press any key to continue . . .

Κώδικας σε C του Apriori αλγορίθμου με εφαρμογή σε μελέτη μετοχών:

Το πρόγραμμα δέχεται ένα αρχείο και το διαβάζει γραμμή γραμμή και με την βοήθεια της strtok σπάει την κάθε γραμμή στα 4 στοιχεία που περιέχει. Στην συνέχεια αποθηκεύονται σε μια δομή και μπαίνουν σε μια απλά συνδεδεμένη λίστα. Δίνουμε από το πληκτρολόγιο τα support και confidence και ξεκινάει το πρόγραμμα βήμα βήμα να υπολογίζει τα support και να εκτυπώνει τα αποτελέσματα. Σε κάθε βήμα τα αποτελέσματα αποθηκεύονται σε ένα νέο struct διαφορετικό για κάθε επίπεδο.
Σε κάθε struct υπάρχουν για κάθε μετοχή 2 μεταβλητές, η πρώτη που περιέχει το όνομα(ή 2αδα ή 3αδα μετοχών) κρατάει το support και η άλλη που είναι του τύπου όνομα_check αποθηκεύει την τιμή «1» αν το support της αντίστοιχης μετοχής είναι μεγαλύτερο από το δοσμένο, αλλιώς αποθηκεύει την τιμή «0». Αυτό θα χρησημοποιειθεί στο επόμενο επίπεδο για να επιλέξουμε ποιοι συνδιασμοί μετοχών είναι αποδεκτοί. Σε κάθε επίπεδο εκτυπώνουμε τα support όλων των μετοχών και μετά τα L, δηλαδή ποιες μετοχές έχουν μεγαλύτερο support από το δοσμένο.
Confidence: Όταν εκτυπώνονται τα L ελέγχουμε αν κάποιο από αυτά είναι κενό. Αν για παράδειγμα είναι κενό το L4 θα βρώ το confidence για το αμέσως προηγούμενο επίπεδο που είναι το L3. Στην συνάρτηση που υπολογίζει το confidence βάζω όλους τους συνδιασμούς που μπορούν να υπάρχουν στο επίπεδο.Otan ektypwnoume ta L elegxw an kapoio apo ta L einai keno. Με την βοήθεια της μεταβλητής ονομα_check που υπάρχει στο κάθε επίπεδο βλέπω ποιοι συνδιασμοί είναι αποδεκτοί και αν έχουν confidence μεγαλύτερο από το δοσμένο. Αν ναι εκτυπώνω τον αντίστοιχο κανόνα.

Εδώ αποθηκεύονται οι μετοχές από το αρχείο. Αν ο αγοραστής έχει αγοράσει την μετοχή θα έχει την τιμή «1» αλλιώς «0».

```
struct agorastis{
  int metoxi1;
  int metoxi2;
  int metoxi3;
  int metoxi4;

  struct agorastis *next;
};
```

Το struct που δημιουργείται για το πρώτο βήμα.
Αποθηκεύω το support κάθε μετοχής.
Κάθε μετοχή έχει μια μεταβλητή τύπου όνομα μετοχής_check όπου γίνεται «1» αν το suport της μετοχής είναι >= από αυτό που δίνεται αλλιώς είναι «0».
Η μεταβλητή cnt_μετοχή μας δίχνει πόσες φορές υπάρχει η μετοχή στο αρχείο.
Το counter μας δίχνει πόσες συνολικά μετοχές της ίδιας εταιρίας έχω στο αρχείο.

```
struct l1{
  float metoxh1;
  int metoxh1_check;
  float cnt_metoxh1;

  float metoxh2;
  int metoxh2_check;
  float cnt_metoxh2;

  float metoxh3;
  int metoxh3_check;
  float cnt_metoxh3;

  float metoxh4;
  int metoxh4_check;
  float cnt_metoxh4;

  float counter;
};
```

Το struct που δημιουργείται για το δεύτερο βήμα.
Αποθηκεύω το support κάθε δυάδας μετοχών.
Κάθε δυάδα έχει μια μεταβλητή τύπου όνομα μετοχής_check όπου γίνεται «1» αν το suport
της δυάδας είναι >= από αυτό που δίνεται αλλιώς είναι «0».

```
struct C2{
  float metoxh1_metoxh2;
  int metoxh1_metoxh2_check;

  float metoxh1_metoxh3;
  int metoxh1_metoxh3_check;

  float metoxh1_metoxh4;
  int metoxh1_metoxh4_check;

  float metoxh2_metoxh3;
  int metoxh2_metoxh3_check;

  float metoxh2_metoxh4;
  int metoxh2_metoxh4_check;

  float metoxh3_metoxh4;
  int metoxh3_metoxh4_check;

};
```

Το struct που δημιουργείται για το τρίτο βήμα.
Αποθηκεύω το support κάθε τριάδας μετοχών.
Κάθε τριάδα έχει μια μεταβλητή τύπου όνομα μετοχής_check όπου γίνεται «1» αν το suport

της τριάδας είναι >= από αυτό που δίνεται αλλιώς είναι «0».

```
struct C3{
  float metoxh1_metoxh2_metoxh3;
  int metoxh1_metoxh2_metoxh3_check;

  float metoxh1_metoxh2_metoxh4;
  int metoxh1_metoxh2_metoxh4_check;

  float metoxh1_metoxh4_metoxh3;
  int metoxh1_metoxh4_metoxh3_check;

  float metoxh3_metoxh4_metoxh2;
  int metoxh3_metoxh4_metoxh2_check;

};
```

Το struct που δημιουργείται για το τέταρτο βήμα.
Αποθηκεύω το support κάθε τετράδας μετοχών.
Κάθε τετράδα έχει μια μεταβλητή τύπου όνομα μετοχής_check όπου γίνεται «1» αν το suport της τετράδας είναι >= από αυτό που δίνεται αλλιώς είναι «0».

```
struct C4{
  float metoxh1_metoxh2_metoxh3_metoxh4;
  int metoxh1_metoxh2_metoxh3_metoxh4_check;
};

void print_list(struct agorastis *);
struct agorastis *insert(struct agorastis *,struct agorastis *);
struct agorastis *make_list_from_file(char *,struct agorastis *);
struct agorastis *new_node(int,int,int,int);

struct l1 create_L1(struct agorastis *,float);
void print_L1(struct l1);

struct C2 create_L2(struct agorastis *,struct l1,float);
void print_L2(struct C2);

struct C3 create_L3(struct C2,struct agorastis *,float,float);
int print_L3(struct C3);

struct C4 create_L4(struct C3,struct agorastis *,float,float);
int print_L4(struct C4);

void Confidence_L2(struct C2,struct l1,float,float);
void Confidence_L3(struct C3, struct C2,struct l1,float,float);
```

Στο argv[1] υπάρχει το όνομα του αρχείου. Στην άσκηση το έχω ονομάσει arxeio1.txt.
Στο argv[0] υπάρχει το εκτελέσιμο της άσκησης (ask1.exe).

```
main(int argc,char *argv[]){

struct agorastis *head=NULL;
struct l1 tmp;
struct C2 tmp1;
struct C3 tmp3;
struct C4 tmp4;
float sup,conf;
float cnt;
int check_if_empty_L3,check_if_empty_L4;
```

Στην περίπτωση μας toargc είναι 2.
Αν βάλουμε λάθος ορίσματα το πρόγραμμα κλείνει.

```
if(argc!=2){
        printf("WRONG number of arguments.\n");
        exit(0);
}
```

Η συνάρτηση δέχεται το αρχείο arxeio1.txt και την κεφαλή της λίστας.
Επιστρέφει την τελική λίστα.

```
head=make_list_from_file(argv[1],head);
```

Εκτυπώνω την λίστα. Ουσιαστικά εκτυπώνω το αρχείο μου.

```
print_list(head);

printf("Dwse support(%%):");
scanf("%f",&sup);
printf("Dwse confidence(%%):");
scanf("%f",&conf);
```

Στην συνέχεια έχω τα 4 βήματα για το sypport.

```
tmp=create_L1(head,sup);
cnt=tmp.counter;
print_L1(tmp);

tmp1=create_L2(head,tmp,sup);
print_L2(tmp1);

tmp3=create_L3(tmp1,head,sup,cnt);
check_if_empty_L3=print_L3(tmp3);

tmp4=create_L4(tmp3,head,sup,cnt);
check_if_empty_L4=print_L4(tmp4);
```

Αν το check_if_empty_L3 είναι 0 σημαίνει το τρίτο βήμα δεν μας δίνει καμία τριάδα με μεγαλύτερο support από αυτό που δώθηκε. Οπότε το l3 είναι άδειο. Ελέγχουμε την εμπιστοσύνη στο αμέσως προηγούμενο βήμα, δηλαδή στο L2.

Ομοίως και για το L4.

```
if(check_if_empty_L3==0){
  Confidence_L2(tmp1,tmp,cnt,conf);
}
else if(check_if_empty_L4==0){
  Confidence_L3(tmp3,tmp1,tmp,cnt,conf);
}
else{;}

system("pause");
}
```

Δέχεται τον κόμβο που έχουμε φτιάξει στην συνάρτηση new_node.
Η συνάρτηση ελέγχει αν η λίστα είναι κενή. Αυτό γίνεται μόνο την πρώτη φορά που γίνεται η πρώτη εισαγωγή. Αν είναι κενή βάζει την κεφαλή της λίστας να δίχνει στον κόμβο αλλιώς διατέχει την λίστα, φτάνει στο τελευταίο στοιχείο της και βάζει τον νέο κόμβο. Τέλος επιστρέφει την κεφαλή της λίστας

```
struct agorastis *insert(struct agorastis *head,struct agorastis *node){

struct agorastis *current=head;

if(head==NULL){
        head=node;
        return head;
}

else{
        while(current->next!=NULL)
                current=current->next;

        current->next=node;
}


        return head;
}
```

Ανοίγω ένα αρχείο και διαβάζω κάθε φορά 1 γραμμή του αρχείου.
Με την strtok παίρνει κάθε φορά ότι υπάρχει μέχρι να βρεί το \t(tab).
Επειδή έχω 4 μετοχές την καλώ 4 φορές.
Μετα καλώ την new_node για να φτίαξω τον κόμβο και μετά την insert για να βάλω τον κόμβο στην λίστα.

```
 struct agorastis *make_list_from_file(char *filename,struct agorastis *head){

struct agorastis *node;
char *s=(char *)malloc(100);
char *temp;
```

```
int met1,met2,met3,met4;
int k=0;
```

Ανοίγω το αρχείο με όνομα filename(εδώ arxeio1.txt) για να το διαβάσω μόνο(r).
Αν είναι άδειο τότε εκτυπώνει μήνυμα λάθους και τερματίζεται το πρόγραμμα.

```
FILE *fp=fopen(filename,"r");
if(fp==NULL){
	printf("Error at opening File %s\n",filename);
	exit(0);
}
```

Αποθηκεύω που παίρνω κάθε φορά στο s με μέγιστο μήκος 100. Δίνω μεγάλο μήκος για να διαβάσει όλη την γραμμή.
Με το k++; μετράω πόσες γραμμές έχει το αρχείο.
Την πρώρη φορά που καλείται η strtok παίρνει ορίσματα το s και το \t ενώ στην συνέχεια πάντα με ορίσματα το NULL και το \t.
Ο έλεγχος «if((temp=strtok(s,"\t"))!=NULL)» γίνεται για να διασφαλίσουμε ότι έχουμε σωστά δεδομένα.

```
while((s=fgets(s,100,fp))!=NULL){
k++;

	if((temp=strtok(s,"\t"))!=NULL)
		met1=atoi(temp);
	else{
		printf("Error1 at reading file.\n");
		exit(0);
	}


	if((temp=strtok(NULL,"\t"))!=NULL)
		met2=atoi(temp);

	else{
		printf("Error2 at reading file.\n");
		exit(0);
	}

	if((temp=strtok(NULL,"\t"))!=NULL)
		met3=atoi(temp);
	else{
		printf("Error3 at reading file.\n");
		exit(0);
	}

	if((temp=strtok(NULL,"\t"))!=NULL)
		met4=atoi(temp);
	else{
		printf("Error4 at reading file.\n");
```

```
                exit(0);
        }
```

Φτίαχνω ένα νέο κόβο – πακέτο με τις 4 μετοχές και στην συνέχεια εισάγω (insert)τον κόμβο στην απλά συνδεδεμένη λίστα.
Απελευθερώνω (free) την μνήμη για το s.
Κλείνω το αρχείο (fclose).
Επιστρέφω την κεφαλή της λίστας(return).

```
        node=new_node(met1,met2,met3,met4);

         head=insert(head,node);
}
free(s);
fclose(fp);
return head;                        }
```

Παίρνω τις 4 μετοχές και φτίαχνω ένα κόμβο.
Τον δείκτη next τον κάνω NULL και επιστρέφω τον κόμβο.

```
struct agorastis *new_node(int met1,int met2,int met3,int met4){

        struct agorastis * tmp=(struct agorastis *)malloc(sizeof(struct agorastis));

        tmp->metoxi1=met1;
        tmp->metoxi2=met2;
        tmp->metoxi3=met3;
        tmp->metoxi4=met4;

        tmp->next=NULL;

return tmp;
}
```

Εκτυπώνω το αρχείο μου (arxeio1.txt).

```
void print_list(struct agorastis *head){

struct agorastis *current=head;
printf("metoxi1  metoxi2  metoxi3  metoxi4\n");
        while(current!=NULL){

printf("   %d     %d      %d     %d \n",current->metoxi1,current->metoxi2,

                                        current- >metoxi3,current->metoxi4);
                current=current->next;
        }
}
```

Κανω το πρώτο βήμα. Βρίσκω το L1 και τα αποθηκεύω σε ένα νέο struct το struct l1.

```
struct l1 create_L1(struct agorastis *head,float support){
      struct agorastis *cur=head;
```

```
      struct l1 temp;
      float counter=0.0;
      float i=0.0,j=0.0,k=0.0,l=0.0;
while(cur!=NULL){
     counter++;
     if(cur->metoxi1==1)     i++;
     if(cur->metoxi2==1)       j++;
     if(cur->metoxi3==1)      k++;
           if(cur->metoxi4==1)       l++;

  cur=cur->next;
}

temp.metoxh3_check=temp.metoxh4_check=temp.metoxh1_check=
temp.metoxh2_check=0;  //arxika to kanw mhden
temp.metoxh1=(i*100.0)/counter;                          //ypologizw to sup(metoxh1)
temp.cnt_metoxh1=i;                              //apo8hkevw poses fores exw thn metoxi1
temp.metoxh2=(j*100.0)/counter;
temp.cnt_metoxh2=j;
temp.metoxh3=(k*100.0)/counter;
temp.cnt_metoxh3=k;
temp.metoxh4=(l*100.0)/counter;
temp.cnt_metoxh4=l;

temp.counter=counter;                                  //genika poses agores exw sto arxeio

printf("\n\n  ***Prwto Vhma***\n");
  if((temp.metoxh1) >= support){                       //elegxw an to s(metoxi1 >=support)
    printf("S(metoxi1)=%f%% >= sup.\n",temp.metoxh1);   //an nai kanw to check =1
    temp.metoxh1_check=1;
  }
  else{
    printf("S(metoxi1)=%f%% < sup.\n",temp.metoxh1);    //alliws kanw to check = 0
    temp.metoxh1_check=0;
  }


  if((temp.metoxh2) >= support){
    printf("S(metoxi2)=%f%% >= sup.\n",temp.metoxh2);
    temp.metoxh2_check=1;
  }
  else{
    printf("S(metoxi2)=%f%% < sup.\n",temp.metoxh2);
    temp.metoxh2_check=0;
  }


  if((temp.metoxh3) >= support){
    printf("S(metoxi3)=%f%% >= sup.\n",temp.metoxh3);
    temp.metoxh3_check=1;
```

```
 }
 else{
   printf("S(metoxi3)=%f%% < sup.\n",temp.metoxh3);
   temp.metoxh3_check=0;
 }


 if((temp.metoxh4) >= support){
   printf("S(metoxi4)=%f%% >= sup.\n",temp.metoxh4);
   temp.metoxh4_check=1;
 }
 else{
  printf("S(metoxi4)=%f%% < sup.\n",temp.metoxh4);
  temp.metoxh4_check=0;
 }

 return temp;
}
```

Εκτυπώνει μόνο όσα από τον παραπάνω έλεγχο έχουν check=1 δηλαδή είναι >= από το δοσμένο support.

```
void print_L1(struct l1 tmp){
    struct l1 temp=tmp;

    printf("\nSynepws L1={");
    if((temp.metoxh1_check)==1)
      printf("metoxi1");
    if((temp.metoxh2_check)==1)
      printf(",metoxi2");
    if((temp.metoxh3_check)==1)
      printf(",metoxi3");
    if((temp.metoxh4_check)==1)
      printf(",metoxi4");

    printf("}.\n\n");
}
```

Περνάω σαν ορίσματα την λίστα, το struct που δημιούργησα στο πρώτο βήμα και το support. Εδώ κάνω το δεύτερο βήμα και αποθηκεύω τα αποτελέσματα στο C2.

```
struct C2 create_L2(struct agorastis *head,struct l1 tmp,float support){
      struct agorastis *cur=head;
      struct l1 t=tmp;
      struct C2 temp;
      float counter=0.0;
      float i0=0.0,i1=0.0,i2=0.0,j0=0.0,j1=0.0,k0=0.0;
while(cur!=NULL){              //elegxw an kai oi 2 metoxes yparxoun se ka8e agorasth
     counter++;                                  //diatrexw thn lista me tis agores
     if(cur->metoxi1==1 && cur->metoxi2==1)       i0++;
     if(cur->metoxi1==1 && cur->metoxi3==1)      i1++;
     if(cur->metoxi1==1 && cur->metoxi4==1)       i2++;
```

```
      if(cur->metoxi2==1   && cur->metoxi3==1)       j0++;
      if(cur->metoxi2==1   && cur->metoxi4==1)       j1++;
      if(cur->metoxi3==1  && cur->metoxi4==1)       k0++;


   cur=cur->next;
}
//apo8hkevw ta suport tis ka8e 2adas metoxwn kai arxikopoiw to check =0
temp.metoxh1_metoxh2=(i0*100.0)/counter;
temp.metoxh1_metoxh2_check=0;
temp.metoxh1_metoxh3=(i1*100.0)/counter;
temp.metoxh1_metoxh3_check=0;
temp.metoxh1_metoxh4=(i2*100.0)/counter;
temp.metoxh1_metoxh4_check=0;
temp.metoxh2_metoxh3=(j0*100.0)/counter;
temp.metoxh2_metoxh3_check=0;
temp.metoxh2_metoxh4=(j1*100.0)/counter;
temp.metoxh2_metoxh4_check=0;
temp.metoxh3_metoxh4=(k0*100.0)/counter;
temp.metoxh3_metoxh4_check=0;


  printf("\n\n  ***Deutero Vhma***\n");

if((t.metoxh1_check==1) && (t.metoxh2_check==1)){

//edw elegxw an tis metoxes exoun perasei apo to prwto vhma..
//dld an sto proto vhma eixan >=support kai epomenws to check
//tous einai 1


  if((temp.metoxh1_metoxh2) >= support){

 //edw elegxw to support gia to 2o vhma an einai >= tote
//to check =1(allo check einai auto...einai check apo to struct
 //tou 2ou vhmatos )
     printf("S(metoxi1,metoxi2)=%f%% >= sup.\n",temp.metoxh1_metoxh2);
     temp.metoxh1_metoxh2_check=1;
  }
  else{
 printf("S(metoxi1,metoxi2)=%f%% < sup
(aporriptetai).\n",temp.metoxh1_metoxh2);
     temp.metoxh1_metoxh2_check=0;          //alliws to kanw 0
  }
}

if((t.metoxh1_check==1) && (t.metoxh3_check==1)){
   if((temp.metoxh1_metoxh3) >= support){
     printf("S(metoxi1,metoxi3)=%f%% >= sup.\n",temp.metoxh1_metoxh3);
     temp.metoxh1_metoxh3_check=1;
  }
```

```
  else{
     printf("S(metoxi1,metoxi3)=%f%% < sup
(aporriptetai).\n",temp.metoxh1_metoxh3);
     temp.metoxh1_metoxh3_check=0;
  }
}

if((t.metoxh1_check==1) && (t.metoxh4_check==1)){
   if((temp.metoxh1_metoxh4) >= support){
     printf("S(metoxi1,metoxi4)=%f%% >= sup.\n",temp.metoxh1_metoxh4);
     temp.metoxh1_metoxh4_check=1;
  }
  else{
     printf("S(metoxi1,metoxi4)=%f%% < sup
(aporriptetai).\n",temp.metoxh1_metoxh4);
     temp.metoxh1_metoxh4_check=0;
  }
}

if((t.metoxh2_check==1) && (t.metoxh3_check==1)){
  if((temp.metoxh2_metoxh3) >= support){
     printf("S(metoxi2,metoxi3)=%f%% >= sup.\n",temp.metoxh2_metoxh3);
     temp.metoxh2_metoxh3_check=1;
  }
  else{
     printf("S(metoxi2,metoxi3)=%f%% < sup
(aporriptetai).\n",temp.metoxh2_metoxh3);
     temp.metoxh2_metoxh3_check=0;
  }
}

if((t.metoxh2_check==1) && (t.metoxh4_check==1)){
  if((temp.metoxh2_metoxh4) >= support){
     printf("S(metoxi2,metoxi4)=%f%% >= sup.\n",temp.metoxh2_metoxh4);
     temp.metoxh2_metoxh4_check=1;
  }
  else{
     printf("S(metoxi2,metoxi4)=%f%% < sup
(aporriptetai).\n",temp.metoxh2_metoxh4);
     temp.metoxh2_metoxh4_check=0;
  }
}

if((t.metoxh3_check==1) && (t.metoxh4_check==1)){
  if((temp.metoxh3_metoxh4) >= support){
     printf("S(metoxi3,metoxi4)=%f%% >= sup.\n",temp.metoxh3_metoxh4);
     temp.metoxh3_metoxh4_check=1;
  }
  else{
```

```
    printf("S(metoxi3,metoxi4)=%f%% < sup
(aporriptetai).\n",temp.metoxh3_metoxh4);
    temp.metoxh3_metoxh4_check=0;
  }
}
return temp;                //epistrefw to struct tou 2ou vhmatos
}

//ektypwnw ta zeugh pou exoun support megalytero apo to dosmeno dld to check tou
2ou vhmatos =1
void print_L2(struct C2 tmp){
    struct C2 temp=tmp;

    printf("\nSynepws L2={\n");

    if((temp.metoxh1_metoxh2_check)==1)
      printf("{metoxi1,metoxi2}\n");
    if((temp.metoxh1_metoxh3_check)==1)
      printf("{metoxi1,metoxi3}\n");
    if((temp.metoxh1_metoxh4_check)==1)
      printf("{metoxi1,metoxi4}\n");
    if((temp.metoxh2_metoxh3_check)==1)
      printf("{metoxi2,metoxi3}\n");
    if((temp.metoxh2_metoxh4_check)==1)
      printf("{metoxi2,metoxi4}\n");
    if((temp.metoxh3_metoxh4_check)==1)
      printf("{metoxi3,metoxi4}\n");

   printf("}\n");
}
```

Ίδια λογική με το προηγούμενο βήμα. Εδώ έχουμε το 3^{ο} βήμα.

```
struct C3 create_L3(struct C2 temp,struct agorastis *head,float support,float counter){
  struct agorastis *cur=head;
  struct C2 tmp=temp;
  struct C3 tmp1;
  float i0=0.0,i1=0.0,i2=0.0,i3=0.0,i4=0.0,i5=0.0,i6=0.0,i7=0.0,i8=0.0,i9=0.0;

tmp1.metoxh1_metoxh2_metoxh3_check=0;
tmp1.metoxh1_metoxh2_metoxh4_check=0;
tmp1.metoxh1_metoxh4_metoxh3_check=0;
tmp1.metoxh3_metoxh4_metoxh2_check=0;

printf("\n\n  ***Trito Vhma***");

if((tmp.metoxh1_metoxh2_check==1 && tmp.metoxh1_metoxh3_check==1) ||
(tmp.metoxh1_metoxh2_check==1 && tmp.metoxh2_metoxh3_check==1) ||
(tmp.metoxh1_metoxh2_check==1 && tmp.metoxh2_metoxh3_check==1))
{
   cur=head;
```

```
  while(cur!=NULL){
                if(cur->metoxi1==1 && cur->metoxi2==1 && cur->metoxi3==1)
            i0++;
                cur=cur->next;
  }
  tmp1.metoxh1_metoxh2_metoxh3=(i0*100.0)/counter;
  if(tmp1.metoxh1_metoxh2_metoxh3>=support){
     tmp1.metoxh1_metoxh2_metoxh3_check=1;
     printf("\nS({metoxi1,metoxi2,metoxi3})= %f >=
support.",tmp1.metoxh1_metoxh2_metoxh3);
   }
   else{
     printf("\nS({metoxi1,metoxi2,metoxi3})= %f <
     support (aporriptetai).",tmp1.metoxh1_metoxh2_metoxh3);
     tmp1.metoxh1_metoxh2_metoxh3_check=0;
   }
}

if((tmp.metoxh1_metoxh2_check==1 && tmp.metoxh1_metoxh4_check==1) ||
(tmp.metoxh1_metoxh2_check==1 && tmp.metoxh2_metoxh4_check==1) ||
(tmp.metoxh2_metoxh4_check==1 && tmp.metoxh1_metoxh4_check==1))
{
  cur=head;
  while(cur!=NULL){
                if(cur->metoxi1==1 && cur->metoxi2==1 && cur->metoxi4==1)
            i1++;
                cur=cur->next;
  }
  tmp1.metoxh1_metoxh2_metoxh4=(i1*100.0)/counter;
  if(tmp1.metoxh1_metoxh2_metoxh4>=support){
     tmp1.metoxh1_metoxh2_metoxh4_check=1;
     printf("\nS({metoxi1,metoxi2,metoxi4})= %f >=
              support.",tmp1.metoxh1_metoxh2_metoxh4);
   }
   else{
     printf("\nS({metoxi1,metoxi2,metoxi4})= %f <
              support (aporriptetai).",tmp1.metoxh1_metoxh2_metoxh4);
     tmp1.metoxh1_metoxh2_metoxh4_check=0;
   }
}

if((tmp.metoxh1_metoxh4_check==1 && tmp.metoxh1_metoxh3_check==1) ||
(tmp.metoxh1_metoxh4_check==1 &&  tmp.metoxh3_metoxh4_check==1) ||
(tmp.metoxh3_metoxh4_check==1 && tmp.metoxh1_metoxh3_check==1))
{
  cur=head;
  while(cur!=NULL){
                if(cur->metoxi1==1 && cur->metoxi4==1 && cur->metoxi3==1)
            i4++;
                cur=cur->next;
```

```
   }
   tmp1.metoxh1_metoxh4_metoxh3=(i4*100.0)/counter;
   if(tmp1.metoxh1_metoxh4_metoxh3>=support){
      tmp1.metoxh1_metoxh4_metoxh3_check=1;
      printf("\nS({metoxi1,metoxi4,metoxi3})= %f >=
                        support.",tmp1.metoxh1_metoxh4_metoxh3);
    }
    else{
      printf("\nS({metoxi1,metoxi4,metoxi3})= %f <
                support (aporriptetai).",tmp1.metoxh1_metoxh4_metoxh3);
      tmp1.metoxh1_metoxh4_metoxh3_check=0;
    }
}

if((tmp.metoxh3_metoxh4_check==1 && tmp.metoxh2_metoxh3_check==1) ||
(tmp.metoxh3_metoxh4_check==1 &&  tmp.metoxh2_metoxh4_check==1) ||
(tmp.metoxh2_metoxh4_check==1 && tmp.metoxh2_metoxh3_check==1))
{
   cur=head;
   while(cur!=NULL){
                    if(cur->metoxi2==1 && cur->metoxi3==1 && cur->metoxi4==1)
               i8++;
                    cur=cur->next;
   }
   tmp1.metoxh3_metoxh4_metoxh2=(i8*100.0)/counter;
   if(tmp1.metoxh3_metoxh4_metoxh2>=support){
      tmp1.metoxh3_metoxh4_metoxh2_check=1;
      printf("\nS({metoxi2,metoxi3,metoxi4})= %f >=
                        support.",tmp1.metoxh3_metoxh4_metoxh2);
    }
    else{
      printf("\nS({metoxi2,metoxi3,metoxi4})= %f <
                support (aporriptetai).",tmp1.metoxh3_metoxh4_metoxh2);
      tmp1.metoxh3_metoxh4_metoxh2_check=0;
      }
}
return tmp1;
}
```

Εκτυπώνω στο L3 μόνο όσες τρυάδες έχουν support >= από αυτό που δώθηκε.
Εδώ υπάρχει μια μεταβλητή checkifempty για να δούμε αν είναι το L3 keno ή όχι.
Βοηθάει στο confidence.

```
int print_L3(struct C3 tmp){
    struct C3 temp=tmp;
    int check_if_empty=0;

    printf("\nSynepws L3={\n");

    if((temp.metoxh1_metoxh2_metoxh3_check)==1){
```

```
        printf("{metoxi1,metoxi2,metoxi3}\n");
        check_if_empty++;
        }
     if((temp.metoxh1_metoxh2_metoxh4_check)==1){
        printf("{metoxi1,metoxi2,metoxi4}\n");
        check_if_empty++;
        }

     if((temp.metoxh1_metoxh4_metoxh3_check)==1){
        printf("{metoxi1,metoxi4,metoxi3}\n");
        check_if_empty++;
        }

     if((temp.metoxh3_metoxh4_metoxh2_check)==1){
        printf("{metoxi2,metoxi3,metoxi4}\n");
        check_if_empty++;
        }

   printf("}\n");
   return check_if_empty;
}
```

Ίδια λογική με τα προηγούμενα.

```
struct C4 create_L4(struct C3 temp,struct agorastis *head,float support,float counter){
  struct agorastis *cur=head;
  struct C3 tmp=temp;
  struct C4 tmp1;
  float i0=0.0;

 tmp1.metoxh1_metoxh2_metoxh3_metoxh4_check=0;
 printf("\n\n  ***Tetarto Vhma***");
if( (tmp.metoxh1_metoxh2_metoxh3_check==1 &&
tmp.metoxh1_metoxh2_metoxh4_check==1) ||
(tmp.metoxh1_metoxh2_metoxh3_check==1 &&
tmp.metoxh1_metoxh4_metoxh3_check==1) ||
(tmp.metoxh1_metoxh2_metoxh3_check==1 &&
tmp.metoxh3_metoxh4_metoxh2_check==1) ||
(tmp.metoxh1_metoxh4_metoxh3_check==1 &&
tmp.metoxh1_metoxh2_metoxh4_check==1) ||
(tmp.metoxh1_metoxh2_metoxh4_check==1 &&
tmp.metoxh3_metoxh4_metoxh2_check==1) ||
(tmp.metoxh1_metoxh4_metoxh3_check==1 &&
tmp.metoxh3_metoxh4_metoxh2_check==1))
{

   cur=head;
   while(cur!=NULL){
                  if(cur->metoxi1==1 && cur->metoxi2==1 && cur->metoxi3==1 &&
cur->metoxi4==1)
```

```
            i0++;
                cur=cur->next;
   }
   tmp1.metoxh1_metoxh2_metoxh3_metoxh4=(i0*100.0)/counter;
    if(tmp1.metoxh1_metoxh2_metoxh3_metoxh4>=support){
       printf("\nS({metoxi1,metoxi2,metoxi3,metoxi4})= %f >=
                 support.\n",tmp1.metoxh1_metoxh2_metoxh3_metoxh4);
       tmp1.metoxh1_metoxh2_metoxh3_metoxh4_check=1;
    }
    else{
      printf("\nS({metoxi1,metoxi2,metoxi3,metoxi4})= %f <
              support (aporriptetai).\n",tmp1.metoxh1_metoxh2_metoxh3_metoxh4);
      tmp1.metoxh1_metoxh2_metoxh3_metoxh4_check=0;
    }
}
return tmp1;
}
int print_L4(struct C4 tmp){
    struct C4 temp=tmp;
    int check_if_empty=0;

    if((temp.metoxh1_metoxh2_metoxh3_metoxh4_check)==1){
      printf("\nSynepws L5={{metoxi1,metoxi2,metoxi3,metoxi4}}.\n");
      check_if_empty++;
      }
 return check_if_empty;
}
```

Υπολογίζει το confidence σε όσες 2άδες έχουν support >= του δοσμένου στο L2
Όταν καλώ την συνάρτηση δίνω το struct του 2^{ου} βήματος, το struct του 1^{ου} βήματος, τον counter και το confidence.

```
void Confidence_L2(struct C2 tmp ,struct l1 tmp1,float counter,float conf){
    struct C2 temp=tmp;
    struct l1 temp1=tmp1;
    float check_conf1,check_conf2;

   printf("\n\n***Check Confidence L2***\n\n");
    //arxika elegxw an h 2ada px edw {metoxi1,metoxi2} exei support >= tou
dosmenou dld check=1
    if((temp.metoxh1_metoxh2_check)==1){
      printf("{metoxi1,metoxi2}\n");

check_conf1=(((temp.metoxh1_metoxh2*counter)/100)/temp1.cnt_metoxh1)*100;
//vriskw confidence gia metoxi1->metoxi2

check_conf2=(((temp.metoxh1_metoxh2*counter)/100)/temp1.cnt_metoxh2)*100;
//vriskw conf gia metoxi2->metoxi1
      if(check_conf1>=conf){              //an auto p vrika gia to prwto einai >=tou
dosmenou cofidence
```

```
        printf("*metoxi1->metoxi2: empistosynh = %f >= confidence
(egrinetai)\n",check_conf1);
        printf("Paragetai o kanonas metoxi1->metoxi2, dhladh opoios agorazei
metoxi1 agorazei kai metoxi2.\n");
      }
      else
        printf("*metoxi1->metoxi2: empistosynh = %f < confidence
(aporiptetai)\n",check_conf1); //alliws....
      if(check_conf2>=conf){                                    //elegxw kai ton
2o kanona
        printf("*metoxi2->metoxi1: empistosynh = %f >= confidence
(egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi2->metoxi1, dhladh opoios agorazei
metoxi2 agorazei kai metoxi1.\n\n");
      }
      else
        printf("*metoxi2->metoxi1: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf2);
    }
    //to idio ginetai opws parapanw kai gia tous ypoloipous syndiasmous
    if((temp.metoxh1_metoxh3_check)==1){
      printf("{metoxi1,metoxi3}\n");

check_conf1=(((temp.metoxh1_metoxh3*counter)/100)/temp1.cnt_metoxh1)*100;

check_conf2=(((temp.metoxh1_metoxh3*counter)/100)/temp1.cnt_metoxh3)*100;
      if(check_conf1>=conf){
        printf("*metoxi1->metoxi3: empistosynh = %f >=
                confidence (egrinetai)\n",check_conf1);
        printf("Paragetai o kanonas metoxi1->metoxi3, dhladh opoios agorazei
metoxi1
                 agorazei kai metoxi3.\n");
      }
      else
        printf("*metoxi1->metoxi3: empistosynh = %f <
              confidence (aporiptetai)\n",check_conf1);
      if(check_conf2>=conf){
        printf("*metoxi3->metoxi1: empistosynh = %f >=
              confidence (egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi3->metoxi1, dhladh opoios agorazei
metoxi3
                 agorazei kai metoxi1.\n\n");
      }
      else
        printf("*metoxi3->metoxi1: empistosynh = %f <
                confidence (aporiptetai)\n\n",check_conf2);
    }

    if((temp.metoxh1_metoxh4_check)==1){
      printf("{metoxi1,metoxi4}\n");
```

```
check_conf1=(((temp.metoxh1_metoxh4*counter)/100)/temp1.cnt_metoxh1)*100;

check_conf2=(((temp.metoxh1_metoxh4*counter)/100)/temp1.cnt_metoxh4)*100;
      if(check_conf1>=conf){
        printf("*metoxi1->metoxi4: empistosynh = %f >=
                  confidence (egrinetai)\n",check_conf1);
        printf("Paragetai o kanonas metoxi1->metoxi4, dhladh opoios agorazei
metoxi1
                  agorazei kai metoxi4.\n");
      }
      else
        printf("*metoxi1->metoxi4: empistosynh = %f <
                  confidence (aporiptetai)\n",check_conf1);
      if(check_conf2>=conf){
        printf("*metoxi4->metoxi1: empistosynh = %f >=
                  confidence (egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi4->metoxi1, dhladh opoios agorazei
metoxi4
                  agorazei kai metoxi1.\n\n");
      }
      else
        printf("*metoxi4->metoxi1: empistosynh = %f <
                 confidence (aporiptetai)\n\n",check_conf2);
    }

    if((temp.metoxh2_metoxh3_check)==1){
      printf("{metoxi2,metoxi3}\n");

check_conf1=(((temp.metoxh2_metoxh3*counter)/100)/temp1.cnt_metoxh2)*100;

check_conf2=(((temp.metoxh2_metoxh3*counter)/100)/temp1.cnt_metoxh3)*100;
      if(check_conf1>=conf){
        printf("*metoxi2->metoxi3: empistosynh = %f >=
                  confidence (egrinetai)\n",check_conf1);
        printf("Paragetai o kanonas metoxi2->metoxi3, dhladh opoios agorazei
metoxi2
                  agorazei kai metoxi3.\n");
      }
      else
        printf("*metoxi2->metoxi3: empistosynh = %f <
                  confidence (aporiptetai)\n",check_conf1);
      if(check_conf2>=conf){
        printf("*metoxi3->metoxi2: empistosynh = %f >=
                  confidence (egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi3->metoxi2, dhladh opoios agorazei
metoxi3
                  agorazei kai metoxi2.\n\n");
      }
      else
```

```
        printf("*metoxi3->metoxi2: empistosynh = %f <
                confidence (aporiptetai)\n\n",check_conf2);
    }

    if((temp.metoxh2_metoxh4_check)==1){
      printf("{metoxi2,metoxi4}\n");

check_conf1=(((temp.metoxh2_metoxh4*counter)/100)/temp1.cnt_metoxh2)*100;

check_conf2=(((temp.metoxh2_metoxh4*counter)/100)/temp1.cnt_metoxh4)*100;
      if(check_conf1>=conf){
        printf("*metoxi2->metoxi4: empistosynh = %f >=
                confidence (egrinetai)\n",check_conf1);
        printf("Paragetai o kanonas metoxi2->metoxi4, dhladh opoios agorazei
metoxi2
                agorazei kai metoxi4.\n");
      }
      else
        printf("*metoxi2->metoxi4: empistosynh = %f <
                confidence (aporiptetai)\n",check_conf1);
      if(check_conf2>=conf){
        printf("*metoxi4->metoxi2: empistosynh = %f >=
                confidence (egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi4->metoxi2, dhladh opoios agorazei
metoxi4
                agorazei kai metoxi2.\n\n");
      }
      else
        printf("*metoxi4->metoxi2: empistosynh = %f <
                confidence (aporiptetai)\n\n",check_conf2);
    }

    if((temp.metoxh3_metoxh4_check)==1){
      printf("{metoxi3,metoxi4}\n");

check_conf1=(((temp.metoxh3_metoxh4*counter)/100)/temp1.cnt_metoxh3)*100;

check_conf2=(((temp.metoxh3_metoxh4*counter)/100)/temp1.cnt_metoxh4)*100;
      if(check_conf1>=conf){
        printf("*metoxi3->metoxi4: empistosynh = %f >=
                confidence (egrinetai)\n",check_conf1);
        printf("Paragetai o kanonas metoxi3->metoxi4, dhladh opoios agorazei
metoxh3
                agorazei kai metoxi4.\n");
      }
      else
        printf("*metoxi3->metoxi4: empistosynh = %f <
                confidence (aporiptetai)\n",check_conf1);
      if(check_conf2>=conf){
        printf("*metoxi4->metoxi3: empistosynh = %f >=
```

```
                confidence (egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi4->metoxi3, dhladh opoios agorazei
metoxi4
                agorazei kai metoxh3.\n\n");
      }
      else
        printf("*metoxi4->metoxi3: empistosynh = %f <
                confidence (aporiptetai)\n\n",check_conf2);
    }


}
```

Ιδια λογική με την παραπάνω.

```
void Confidence_L3(struct C3 tmp, struct C2 t,struct l1 tmp1,float counter,float
conf){
    struct C3 temp=tmp;
    struct l1 temp1=tmp1;
     struct C2 t2=t;
    float
check_conf1,check_conf2,check_conf3,check_conf4,check_conf5,check_conf6;

   printf("\n\n***Check Confidence L3***\n\n");

  if((temp.metoxh1_metoxh2_metoxh3_check)==1){
     printf("{metoxi1,metoxi2,metoxi3}\n");
       check_conf1=(  ((temp.metoxh1_metoxh2_metoxh3*counter)/100)  /
((t2.metoxh1_metoxh2*counter)/100)  )*100;

check_conf2=(((temp.metoxh1_metoxh2_metoxh3*counter)/100)/temp1.cnt_metoxh
3)*100;
       check_conf3=(  ((temp.metoxh1_metoxh2_metoxh3*counter)/100)  /
((t2.metoxh1_metoxh3*counter)/100)  )*100;

check_conf4=(((temp.metoxh1_metoxh2_metoxh3*counter)/100)/temp1.cnt_metoxh
2)*100;
       check_conf5=(  ((temp.metoxh1_metoxh2_metoxh3*counter)/100)  /
((t2.metoxh2_metoxh3*counter)/100)  )*100;

check_conf6=(((temp.metoxh1_metoxh2_metoxh3*counter)/100)/temp1.cnt_metoxh
1)*100;
       if(check_conf1>=conf){
         printf("*{metoxi1->metoxi2}->metoxi3: empistosynh = %f >=
                confidence (egrinetai)\n",check_conf1);
         printf("Paragetai o kanonas {metoxi1->metoxi2}->metoxi3, dhladh opoios
agorazei
                metoxi1 kai metoxi2 agorazei kai metoxh3.\n");
       }
       else
```

```
        printf("*{metoxi1->metoxi2}->metoxi3: empistosynh = %f <
                confidence (aporiptetai)\n",check_conf1);

      if(check_conf2>=conf){
        printf("*metoxi3->{metoxi1->metoxi2}: empistosynh = %f >= confidence
                (egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi3->{metoxi1->metoxi2}, dhladh opoios
agorazei
                 metoxh3 agorazei metoxi1 kai metoxi2.\n\n");
      }
      else
        printf("*metoxi3->{metoxi1->metoxi2}: empistosynh = %f < confidence
                 (aporiptetai)\n\n",check_conf2);

    if(check_conf3>=conf){
        printf("*{metoxi1->metoxi3}->metoxi2: empistosynh = %f >=
               confidence(egrinetai)\n",check_conf3);
        printf("Paragetai o kanonas {metoxi1->metoxi3}->metoxi2, dhladh opoios
agorazei
                 metoxi1 kai metoxh3 agorazei kai metoxi2.\n");
      }
      else
        printf("*{metoxi1->metoxi3}->metoxi2: empistosynh = %f <
                confidence (aporiptetai)\n",check_conf3);


      if(check_conf4>=conf){
        printf("*metoxi2->{metoxi1->metoxi3}: empistosynh = %f >=
                 confidence (egrinetai)\n",check_conf4);
        printf("Paragetai o kanonas metoxi2->{metoxi1->metoxi3}, dhladh opoios
agorazei
                 metoxi2 agorazei metoxi1 kai metoxh3.\n\n");
      }
      else
        printf("*metoxi2->{metoxi1->metoxi3}: empistosynh = %f <
                confidence (aporiptetai)\n\n",check_conf4);

      if(check_conf5>=conf){
        printf("*{metoxi2->metoxi3}->metoxi1: empistosynh = %f >=
                 confidence (egrinetai)\n",check_conf5);
        printf("Paragetai o kanonas {metoxi2->metoxi3}->metoxi1, dhladh opoios
agorazei
                 metoxi2 kai metoxh3 agorazei kai metoxi1.\n");
      }
      else
        printf("*{metoxi2->metoxi3}->metoxi1: empistosynh = %f <
                 confidence (aporiptetai)\n",check_conf5);

      if(check_conf6>=conf){
        printf("*metoxi1->{metoxi2->metoxi3}: empistosynh = %f >=
```

```
                    confidence (egrinetai)\n",check_conf6);
          printf("Paragetai o kanonas metoxi1->{metoxi2->metoxi3}, dhladh opoios
agorazei
                     metoxi1 agorazei metoxi2 kai metoxh3.\n\n");
        }
        else
          printf("*metoxi1->{metoxi2->metoxi3}: empistosynh = %f <
                  confidence (aporiptetai)\n\n",check_conf6);
 }


  if((temp.metoxh1_metoxh2_metoxh4_check)==1){
     printf("{metoxi1,metoxi2,metoxi4}\n");
       check_conf1=(  ((temp.metoxh1_metoxh2_metoxh4*counter)/100)  /
((t2.metoxh1_metoxh2*counter)/100)  )*100;

check_conf2=(((temp.metoxh1_metoxh2_metoxh4*counter)/100)/temp1.cnt_metoxh
4)*100;
       check_conf3=(  ((temp.metoxh1_metoxh2_metoxh4*counter)/100)  /
((t2.metoxh1_metoxh4*counter)/100)  )*100;

check_conf4=(((temp.metoxh1_metoxh2_metoxh4*counter)/100)/temp1.cnt_metoxh
2)*100;
       check_conf5=(  ((temp.metoxh1_metoxh2_metoxh4*counter)/100)  /
((t2.metoxh2_metoxh4*counter)/100)  )*100;

check_conf6=(((temp.metoxh1_metoxh2_metoxh4*counter)/100)/temp1.cnt_metoxh
1)*100;
       if(check_conf1>=conf){
          printf("*{metoxi1->metoxi2}->metoxi4: empistosynh = %f >=
                    confidence (egrinetai)\n",check_conf1);
          printf("Paragetai o kanonas {metoxi1->metoxi2}->metoxi4, dhladh opoios
agorazei
           metoxi1 kai metoxi2 agorazei kai metoxi4.\n");
       }
       else
          printf("*{metoxi1->metoxi2}->metoxi4: empistosynh = %f < confidence
(aporiptetai)\n",check_conf1);

       if(check_conf2>=conf){
          printf("*metoxi4->{metoxi1->metoxi2}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf2);
          printf("Paragetai o kanonas metoxi4->{metoxi1->metoxi2}, dhladh opoios
agorazei metoxi4 agorazei metoxi1 kai metoxi2.\n\n");
       }
       else
          printf("*metoxi4->{metoxi1->metoxi2}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf2);

     if(check_conf3>=conf){
```

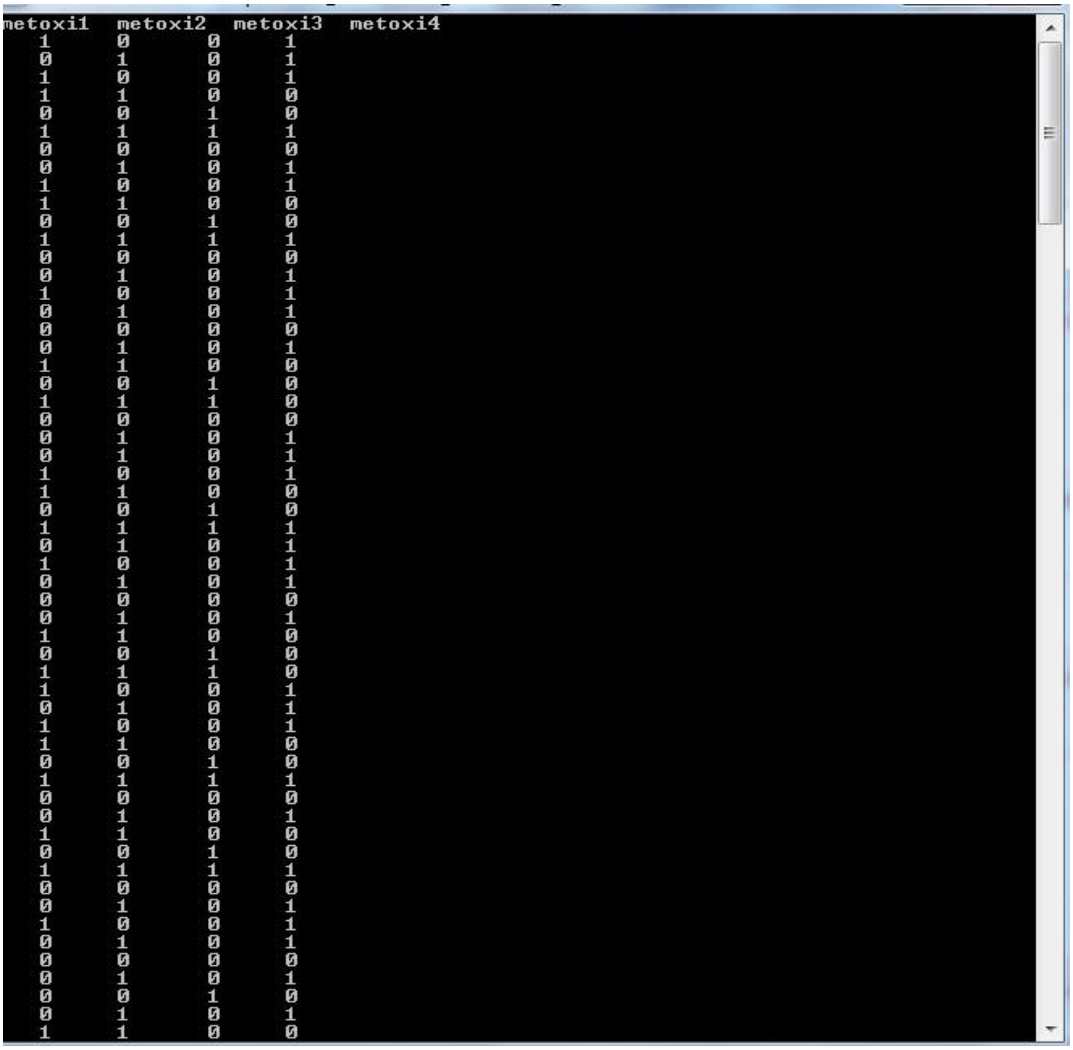
```
        printf("*{metoxi1->metoxi4}->metoxi2: empistosynh = %f >= confidence
(egrinetai)\n",check_conf3);
        printf("Paragetai o kanonas {metoxi1->metoxi4}->metoxi2, dhladh opoios
agorazei metoxi1 kai metoxi4 agorazei kai metoxi2.\n");
      }
      else
        printf("*{metoxi1->metoxi4}->metoxi2: empistosynh = %f < confidence
(aporiptetai)\n",check_conf3);


      if(check_conf4>=conf){
        printf("*metoxi2->{metoxi1->metoxi4}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf4);
        printf("Paragetai o kanonas metoxi2->{metoxi1->metoxi4}, dhladh opoios
agorazei metoxi2 agorazei metoxi1 kai metoxi4.\n\n");
      }
      else
        printf("*metoxi2->{metoxi1->metoxi4}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf4);

      if(check_conf5>=conf){
        printf("*{metoxi2->metoxi4}->metoxi1: empistosynh = %f >= confidence
(egrinetai)\n",check_conf5);
        printf("Paragetai o kanonas {metoxi2->metoxi4}->metoxi1, dhladh opoios
agorazei metoxi2 kai metoxi4 agorazei kai metoxi1.\n");
      }
      else
        printf("*{metoxi2->metoxi4}->metoxi1: empistosynh = %f < confidence
(aporiptetai)\n",check_conf5);

      if(check_conf6>=conf){
        printf("*metoxi1->{metoxi2->metoxi4}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf6);
        printf("Paragetai o kanonas metoxi1->{metoxi2->metoxi4}, dhladh opoios
agorazei metoxi1 agorazei metoxi2 kai metoxi4.\n\n");
      }
      else
        printf("*metoxi1->{metoxi2->metoxi4}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf6);
 }


  if((temp.metoxh1_metoxh4_metoxh3_check)==1){
     printf("{metoxi1,metoxi4,metoxi3}\n");
       check_conf1=(  ((temp.metoxh1_metoxh4_metoxh3*counter)/100)  /
((t2.metoxh1_metoxh4*counter)/100)  )*100;

check_conf2=(((temp.metoxh1_metoxh4_metoxh3*counter)/100)/temp1.cnt_metoxh
3)*100;
```

```
	check_conf3=(  ((temp.metoxh1_metoxh4_metoxh3*counter)/100)  /
((t2.metoxh1_metoxh3*counter)/100)  )*100;

check_conf4=(((temp.metoxh1_metoxh4_metoxh3*counter)/100)/temp1.cnt_metoxh
4)*100;
	check_conf5=(  ((temp.metoxh1_metoxh4_metoxh3*counter)/100)  /
((t2.metoxh3_metoxh4*counter)/100)  )*100;

check_conf6=(((temp.metoxh1_metoxh4_metoxh3*counter)/100)/temp1.cnt_metoxh
1)*100;
	if(check_conf1>=conf){
	  printf("*{metoxi1->metoxi4}->metoxi3: empistosynh = %f >= confidence
(egrinetai)\n",check_conf1);
	  printf("Paragetai o kanonas {metoxi1->metoxi4}->metoxi3, dhladh opoios
agorazei metoxi1 kai metoxi4 agorazei kai metoxh3.\n");
	}
	else
	  printf("*{metoxi1->metoxi4}->metoxi3: empistosynh = %f < confidence
(aporiptetai)\n",check_conf1);

	if(check_conf2>=conf){
	  printf("*metoxi3->{metoxi1->metoxi4}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf2);
	  printf("Paragetai o kanonas metoxi3->{metoxi1->metoxi4}, dhladh opoios
agorazei metoxh3 agorazei metoxi1 kai metoxi4.\n\n");
	}
	else
	  printf("*metoxi3->{metoxi1->metoxi4}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf2);

     if(check_conf3>=conf){
	  printf("*{metoxi1->metoxi3}->metoxi4: empistosynh = %f >= confidence
(egrinetai)\n",check_conf3);
	  printf("Paragetai o kanonas {metoxi1->metoxi3}->metoxi4, dhladh opoios
agorazei metoxi1 kai metoxh3 agorazei kai metoxi4.\n");
	}
	else
	  printf("*{metoxi1->metoxi3}->metoxi4: empistosynh = %f < confidence
(aporiptetai)\n",check_conf3);


	if(check_conf4>=conf){
	  printf("*metoxi4->{metoxi1->metoxi3}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf4);
	  printf("Paragetai o kanonas metoxi4->{metoxi1->metoxi3}, dhladh opoios
agorazei metoxi4 agorazei metoxi1 kai metoxh3.\n\n");
	}
	else
	  printf("*metoxi4->{metoxi1->metoxi3}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf4);
```

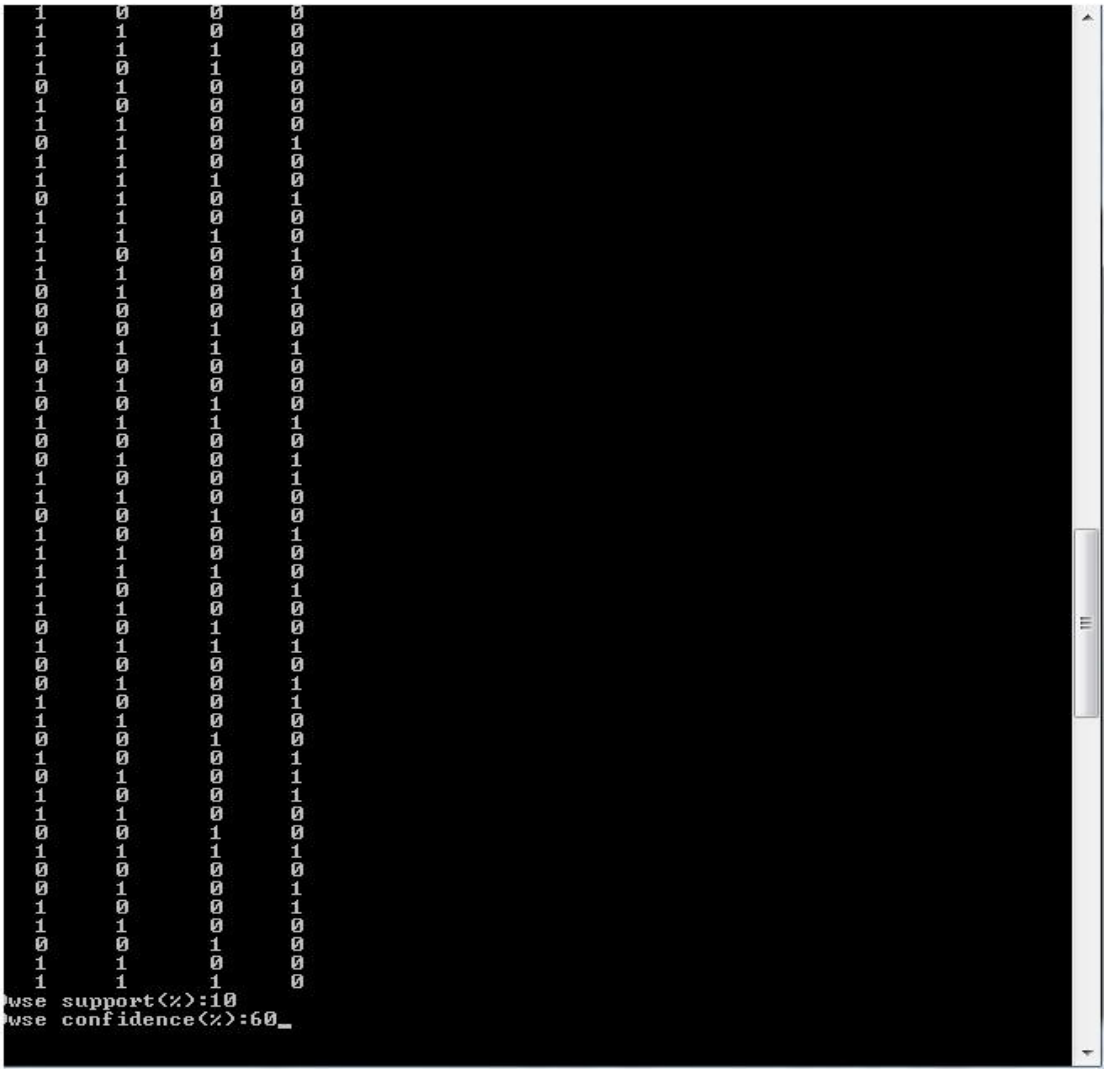
```
        if(check_conf5>=conf){
          printf("*{metoxi3->metoxi4}->metoxi1: empistosynh = %f >= confidence
(egrinetai)\n",check_conf5);
          printf("Paragetai o kanonas {metoxi3->metoxi4}->metoxi1, dhladh opoios
agorazei metoxi4 kai metoxh3 agorazei kai metoxi1.\n");
        }
        else
          printf("*{metoxi3->metoxi4}->metoxi1: empistosynh = %f < confidence
(aporiptetai)\n",check_conf5);

        if(check_conf6>=conf){
          printf("*metoxi1->{metoxi3->metoxi4}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf6);
          printf("Paragetai o kanonas metoxi1->{metoxi3->metoxi4}, dhladh opoios
agorazei metoxi1 agorazei metoxi4 kai metoxh3.\n\n");
        }
        else
          printf("*metoxi1->{metoxi3->metoxi4}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf6);
 }

  if((temp.metoxh3_metoxh4_metoxh2_check)==1){
     printf("{metoxi2,metoxi3,metoxi4}\n");
        check_conf1=(  ((temp.metoxh3_metoxh4_metoxh2*counter)/100)  /
((t2.metoxh2_metoxh3*counter)/100)  )*100;

check_conf2=(((temp.metoxh3_metoxh4_metoxh2*counter)/100)/temp1.cnt_metoxh
4)*100;
        check_conf3=(  ((temp.metoxh3_metoxh4_metoxh2*counter)/100)  /
((t2.metoxh2_metoxh4*counter)/100)  )*100;

check_conf4=(((temp.metoxh3_metoxh4_metoxh2*counter)/100)/temp1.cnt_metoxh
3)*100;
        check_conf5=(  ((temp.metoxh3_metoxh4_metoxh2*counter)/100)  /
((t2.metoxh3_metoxh4*counter)/100)  )*100;

check_conf6=(((temp.metoxh3_metoxh4_metoxh2*counter)/100)/temp1.cnt_metoxh
2)*100;
        if(check_conf1>=conf){
          printf("*{metoxi2->metoxi3}->metoxi4: empistosynh = %f >= confidence
(egrinetai)\n",check_conf1);
          printf("Paragetai o kanonas {metoxi2->metoxi3}->metoxi4, dhladh opoios
agorazei metoxh3 kai metoxi2 agorazei kai metoxi4.\n");
        }
        else
          printf("*{metoxi2->metoxi3}->metoxi4: empistosynh = %f < confidence
(aporiptetai)\n",check_conf1);

        if(check_conf2>=conf){
```

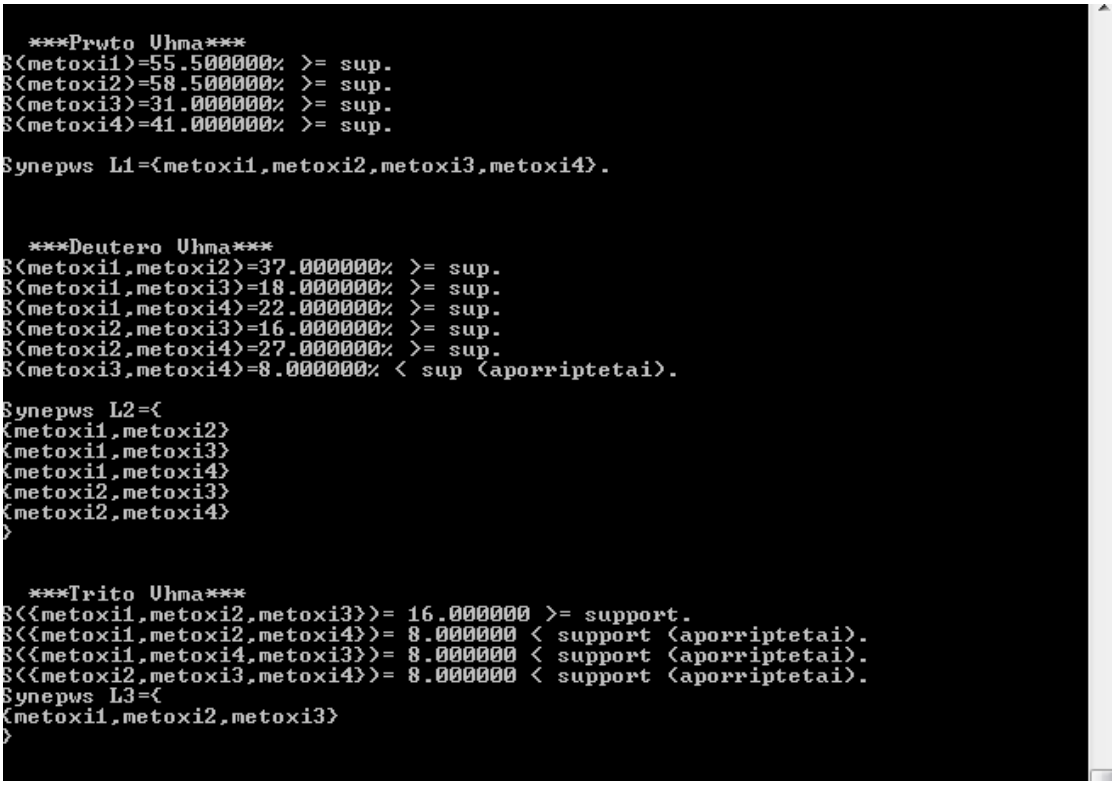
```
        printf("*metoxi4->{metoxi2->metoxi3}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf2);
        printf("Paragetai o kanonas metoxi4->{metoxi2->metoxi3}, dhladh opoios
agorazei metoxi4 agorazei metoxh3 kai metoxi2.\n\n");
      }
      else
        printf("*metoxi4->{metoxi2->metoxi3}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf2);

    if(check_conf3>=conf){
        printf("*{metoxi2->metoxi4}->metoxi3: empistosynh = %f >= confidence
(egrinetai)\n",check_conf3);
        printf("Paragetai o kanonas {metoxi2->metoxi4}->metoxi3, dhladh opoios
agorazei metoxi2 kai metoxi4 agorazei kai metoxh3.\n");
      }
      else
        printf("*{metoxi2->metoxi4}->metoxi3: empistosynh = %f < confidence
(aporiptetai)\n",check_conf3);


      if(check_conf4>=conf){
        printf("*metoxi3->{metoxi2->metoxi4}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf4);
        printf("Paragetai o kanonas metoxi3->{metoxi2->metoxi4}, dhladh opoios
agorazei metoxh3 agorazei metoxi2 kai metoxi4.\n\n");
      }
      else
        printf("*metoxi3->{metoxi2->metoxi4}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf4);

      if(check_conf5>=conf){
        printf("*{metoxi3->metoxi4}->metoxi2: empistosynh = %f >= confidence
(egrinetai)\n",check_conf5);
        printf("Paragetai o kanonas {metoxi3->metoxi4}->metoxi2, dhladh opoios
agorazei metoxh3 kai metoxi4 agorazei kai metoxi2.\n");
      }
      else
        printf("*{metoxi3->metoxi4}->metoxi2: empistosynh = %f < confidence
(aporiptetai)\n",check_conf5);

      if(check_conf6>=conf){
        printf("*metoxi2->{metoxi3->metoxi4}: empistosynh = %f >= confidence
(egrinetai)\n",check_conf6);
        printf("Paragetai o kanonas metoxi2->{metoxi3->metoxi4}, dhladh opoios
agorazei metoxi2 agorazei metoxh3 kai metoxi4.\n\n");
      }
      else
        printf("*metoxi2->{metoxi3->metoxi4}: empistosynh = %f < confidence
(aporiptetai)\n\n",check_conf6);
 }
```

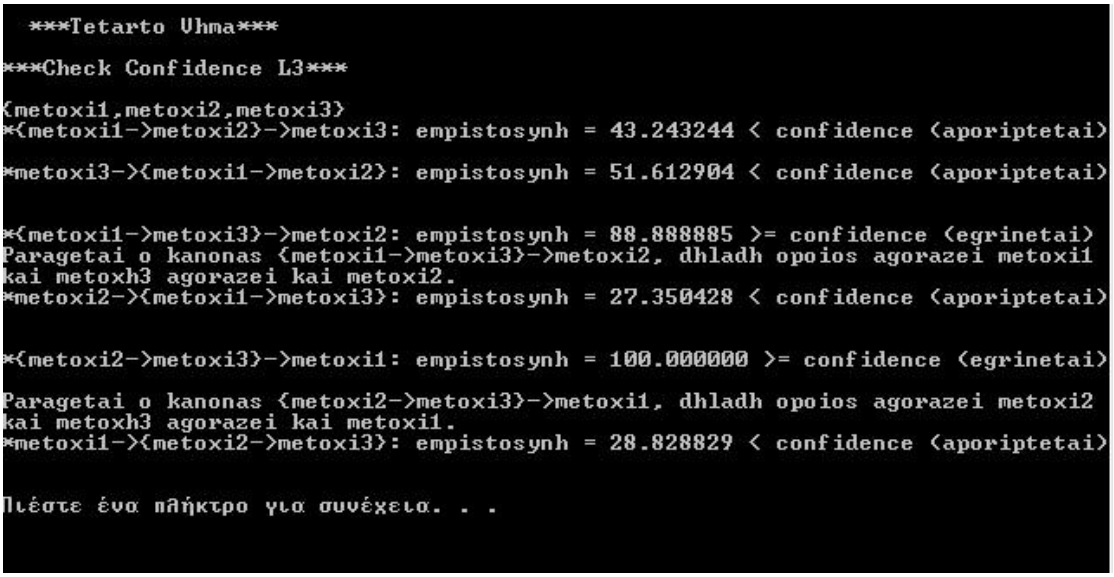
Στηγμιότυπο από το τρέξιμο του αλγορίθμου στον dev:



```
metoxi1  metoxi2  metoxi3  metoxi4
   1        0        0        1
   0        1        0        1
   1        0        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        1
   0        0        0        0
   0        1        0        1
   1        0        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        1
   0        0        0        0
   0        1        0        1
   1        0        0        1
   0        1        0        1
   0        0        0        0
   0        1        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        0
   0        0        0        0
   0        1        0        1
   0        1        0        1
   1        0        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        1
   0        1        0        1
   1        0        0        1
   0        1        0        1
   0        0        0        0
   0        1        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        0
   1        0        0        1
   0        1        0        1
   1        0        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        1
   0        0        0        0
   0        1        0        1
   1        1        0        0
   0        0        1        0
   1        1        1        1
   0        0        0        0
   0        1        0        1
   1        0        0        1
   0        1        0        1
   0        0        0        0
   0        1        0        1
   0        0        1        0
   0        1        0        1
   1        1        0        0
```


1 0 0 0
1 1 0 0
1 1 1 0
1 0 1 0
0 1 0 0
1 0 0 0
1 1 0 0
0 1 0 1
1 1 0 0
1 1 1 0
0 1 0 1
1 1 0 0
1 1 1 0
1 0 0 1
1 1 0 0
0 1 0 1
0 0 0 0
0 0 1 0
1 1 1 1
0 0 0 0
1 1 0 0
0 0 1 0
1 1 1 1
0 0 0 0
0 1 0 1
1 0 0 1
1 1 0 0
0 0 1 0
1 0 0 1
1 1 0 0
1 1 1 0
1 0 0 1
1 1 0 0
0 0 1 0
1 1 1 1
0 0 0 0
0 1 0 1
1 0 0 1
1 1 0 0
0 0 1 0
1 0 0 1
0 1 0 1
1 0 0 1
1 1 0 0
0 0 1 0
1 1 1 1
0 0 0 0
0 1 0 1
1 0 0 1
1 1 0 0
0 0 1 0
1 1 0 0
1 1 1 0
wse support(%):10
wse confidence(%):60


Prwto Vhma
S(metoxi1)=55.500000% >= sup.
S(metoxi2)=58.500000% >= sup.
S(metoxi3)=31.000000% >= sup.
S(metoxi4)=41.000000% >= sup.
Synepws L1={metoxi1,metoxi2,metoxi3,metoxi4}.
Deutero Vhma
S(metoxi1,metoxi2)=37.000000% >= sup.
S(metoxi1,metoxi3)=18.000000% >= sup.
S(metoxi1,metoxi4)=22.000000% >= sup.
S(metoxi2,metoxi3)=16.000000% >= sup.
S(metoxi2,metoxi4)=27.000000% >= sup.
S(metoxi3,metoxi4)=8.000000% < sup (aporriptetai).
Synepws L2={
{metoxi1,metoxi2}
{metoxi1,metoxi3}
{metoxi1,metoxi4}
{metoxi2,metoxi3}
{metoxi2,metoxi4}
}
Trito Vhma
S({metoxi1,metoxi2,metoxi3})= 16.000000 >= support.
S({metoxi1,metoxi2,metoxi4})= 8.000000 < support (aporriptetai).
S({metoxi1,metoxi4,metoxi3})= 8.000000 < support (aporriptetai).
S({metoxi2,metoxi3,metoxi4})= 8.000000 < support (aporriptetai).
Synepws L3={
{metoxi1,metoxi2,metoxi3}
}


Tetarto Vhma
Check Confidence L3
{metoxi1,metoxi2,metoxi3}
*{metoxi1->metoxi2}->metoxi3: empistosynh = 43.243244 < confidence (aporiptetai)
*metoxi3->{metoxi1->metoxi2}: empistosynh = 51.612904 < confidence (aporiptetai)
*{metoxi1->metoxi3}->metoxi2: empistosynh = 88.888885 >= confidence (egrinetai)
Paragetai o kanonas {metoxi1->metoxi3}->metoxi2, dhladh opoios agorazei metoxi1
kai metoxh3 agorazei kai metoxi2.
*metoxi2->{metoxi1->metoxi3}: empistosynh = 27.350428 < confidence (aporiptetai)
*{metoxi2->metoxi3}->metoxi1: empistosynh = 100.000000 >= confidence (egrinetai)
Paragetai o kanonas {metoxi2->metoxi3}->metoxi1, dhladh opoios agorazei metoxi2
kai metoxh3 agorazei kai metoxi1.
*metoxi1->{metoxi2->metoxi3}: empistosynh = 28.828829 < confidence (aporiptetai)
Πιέστε ένα πλήκτρο για συνέχεια. . .