ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Πτυχιακή Εργασία

Bagging και Boosting Μέθοδοι για την Κατασκευή Μοντέλων με Εφαρμογές στα Χρηματοοικονομικά

ΟΛΥΜΠΙΟΣ ΣΩΚΡΑΤΗΣ

Επιβλέπων: Κουκουβίνος Χρήστος, Καθηγητής Ε.Μ.Π.

Αθήνα , Ιούλιος 2013

"All knowledge is, in final analysis history.

All sciences are, in the abstract, mathematics.

All judgments are, in the rationale, statistics"

C. R. Rao

Περίληψη

Το πλήθος των πληροφοριών που καλούμαστε στις μέρες μας να επεξεργαστούμε και η αποτυχία των “κλασσικών” μεθόδων στατιστικής στην ανάλυση και εξαγωγή χρήσιμων πληροφοριών από μεγάλα σετ δεδομένων οδήγησε στην ανάπτυξη της Εξόρυξης Δεδομένων. Η Εξόρυξη Δεδομένων (Data Mining) είναι μια σειρά από τεχνικές που χρησιμοποιούνται πλέον σχεδόν σε όλους του τομείς της σύγχρονης κοινωνίας (ιατρική, οικονομικά, marketing κτλ). Επιπλέον τα τελευταία χρόνια παρουσιάζεται έντονο ενδιαφέρον στις Bagging και Boosting μεθόδους οι οποίες όπως θα δούμε στη συνέχεια ενισχύουν τους αλγόριθμους παρέχοντας μας σαφώς βελτιωμένες προβλέψεις.

Στο πρώτο κεφάλαιο παρουσιάζεται εκτενώς η έννοια του Data Mining, παρουσιάζοντας τις βασικές κατηγορίες του και εξηγώντας, βήμα προς βήμα, τη διαδικασία που ακολουθείται στο data mining. Στο τέλος του πρώτου κεφαλαίου παρουσιάζονται κάποιες εφαρμογές του Data Mining, δίνοντας έμφαση στις εφαρμογές του στον τομέα των Οικονομικών.

Στο δεύτερο κεφάλαιο αναλύονται διάφορες τεχνικές εξόρυξης γνώσης που θα χρησιμοποιηθούν στο τέταρτο κεφάλαιο στην εφαρμογή μας. Συγκεκριμένα παρουσιάζονται τα *Τεχνητά Νευρωνικά Δίκτυα* με έμφαση στο MultiLayer Perceptron (MLP), η *Λογιστική Παλινδρόμηση* με τις μεθόδους εκτίμησης συντελεστών, μέγιστης πιθανοφάνειας και έλεγχο σημαντικότητας. Επιπλέον παρουσιάζεται ένα παράδειγμα λογιστικής παλινδρόμησης στην R και τέλος *τα δέντρα αποφάσεων* καθώς και πως επιλέγουμε τα χαρακτηριστικά τους και με ποια κριτήρια διακόπτουμε και κλαδεύουμε τα δέντρα μας.

Στο τρίτο κεφάλαιο παρουσιάζονται οι μέθοδοι του *Bagging* και *Boosting*, εξηγώντας πλήρως την λογική πίσω από αυτές τις μεθόδους, που όπως θα δούμε είναι γοητευτικά απλή. Στη συνέχεια παρουσιάζονται οι μέθοδοι και αλγόριθμοι των Rainforest και Adaboost για το Bagging και Boosting, αντίστοιχα, καθώς και κάποιες πρόσφατες επεκτάσεις του Adaboost.

Στο τέταρτο και τελευταίο κεφάλαιο επιχειρούμε την εφαρμογή και χρήση και των παραπάνω αλγορίθμων για την πρόβλεψη της κίνησης των μετοχών χρησιμοποιώντας το στατιστικό πρόγραμμα R. Πιο συγκεκριμένα, αφού αναλύσουμε τα δεδομένα μας ως χρονοσειρά, χρησιμοποιούμε τις μεθόδους της Λογιστικής Παλινδρόμησης, MLP, Rainforest, realBoosting και gentleBoosting για να προβλέψουμε την μελλοντική τάση της μετοχής μας. Τα αποτελέσματα που παίρνουμε είναι πολύ ενδιαφέροντα και ικανοποιητικά αφού μας δίνουν μια μεγάλη ακρίβεια πρόβλεψης.

Abstract

The sheer number of information we are called to process and the failure of the “classic” statistical methods in analyzing and extracting useful information from big data sets led to the development of Data Mining. Data Mining is a series of technics that in our days are used in almost every sector of the society (medicine, finance, marketing etc.). Furthermore the last years an intense interest is shown in Bagging and Boosting methods, which enhance the algorithms providing us with clearly improved predictions.

In the first chapter the concept of Data mining is presented extensively, showing its main categories and explaining, step by step, the process that is followed in data mining. In the end of the first chapter we illustrate some applications of data mining, focusing in the financial area applications.

In the second chapter some extraction knowledge technics are analyzed, that will be used later on for our financial application in the fourth chapter. The *Artificial Neural Networks* are presented, emphasizing on the MultiLayer Perceptron (MLP) algorithm, *Logistic Regression* with methods for predicting the coefficients, maximum likelihood, significance tests and an example of logistic regression in R. Finally, we present *Decision trees* and methods to select the features of the trees, ending criteria and pruning methods.

In the third chapter the methods of Bagging and Boosting are presented, explaining in depth the rationale behind this methods, which is “elegantly” simple. Continuing the methods and algorithms of Rainforests and Adaboost are presented with some recent upgrades in the AdaBoost algorithm.

In the fourth and final chapter we attempt the application and use of the above mentioned algorithms in predicting future stock movement using the statistical package R. Specifically, after analyzing our data as time series, we use Logistic Regression, MLP, Rainforests, realAdaBoost and gentleAdaBoost to predict the future trend of our stocks. The results are not only interesting but give us a high prediction accuracy.

Ευχαριστίες

Πρωτίστως αισθάνομαι την ανάγκη να ευχαριστήσω εκ βάθους καρδίας τον Καθηγητή του Ε.Μ.Π. κ. Χρήστο Κουκουβίνο, όχι μόνο για τη δυνατότητα που μου προσέφερε να ασχοληθώ με αυτό το πολύ ενδιαφέρον θέμα, αλλά και για τη συνεχή του προσφορά καθ' όλη τη διάρκεια των σπουδών μου που με οδήγησε τελικά στο να αγαπήσω τη στατιστική και να επιλέξω τον τομέα αυτό για τη συνέχεια των σπουδών μου.

Ιδιαίτερες ευχαριστίες θα ήθελα να εκφράσω στην υποψήφια διδάκτορα Χριστίνα Παρπούλα, για την πολύτιμη βοήθεια της και το συνεχές ενδιαφέρον κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Ακόμα θα ήθελα να ευχαριστήσω τους γονείς μου Ανδρέα και Μαρία Ολυμπίου και την αδερφή μου Δέσποινα για την αμέριστη υπομονή τους και άμεση υποστήριξή τους καθ' όλη τη διάρκεια των σπουδών μου. Τέλος θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου, όχι μόνο για την βοήθεια και συμπαράσταση που μου έδωσαν κατά τη διάρκεια αυτής της εργασίας, αλλά και για τα πέντε υπέροχα χρόνια που πέρασα μαζί τους ως φοιτητής του Ε.Μ.Π.

Ολύμπιος Σωκράτης

Αθήνα, 2013

Περιεχόμενα

Κεφάλαιο 1. : Data mining

1.1 Εισαγωγή

Από πολύ παλιά διαφάνηκε η ανάγκη εξόρυξης μοτίβων και πληροφοριών από δεδομένα. Οι προσπάθειες εξόρυξης πληροφοριών ξεκίνησαν γύρω στο 1700 με το γνωστό θεώρημα του Bayes και συνεχίστηκαν το 1800 με την ανάπτυξη της ανάλυσης παλινδρόμησης. Αυτές οι μέθοδοι ήταν αρκετές για την ανάλυση δεδομένων μέχρι τα μέσα του 1900 οπότε και άρχισε η ραγδαία ανάπτυξη της πληροφορικής και της τεχνολογίας γενικότερα.

Αυτή η ραγδαία ανάπτυξη της πληροφορικής κατέστησε εύκολη τη συλλογή και αποθήκευση δεδομένων και συγχρόνως παρουσιάστηκε η εμφανής αδυναμία των μέχρι τότε γνωστών μεθόδων στατιστικής να αναλύσουν αυτά τα μεγάλα σετ δεδομένων. Λόγω αυτής της αδυναμίας άρχισε να αναπτύσσεται η εξόρυξη δεδομένων (data mining) γύρω στο 1990.

Το data mining έχει ως στόχο την εξαγωγή χρήσιμων μοτίβων και συμπερασμάτων από πολύ μεγάλα σετ δεδομένων. Το επιτυγχάνει αυτό συνδυάζοντας την επιστήμη της στατιστικής και της πληροφορικής, δηλαδή με τη χρησιμοποίηση μεθόδων που βασίζονται στην τεχνητή νοημοσύνη και εκμάθηση μηχανής.

Αν και δεν υπάρχει ένας κοινός αποδεχτώς ορισμός για το data mining θα δεχτούμε τον ορισμό των Han και Kamber *"Πολλές τεχνικές που μπορούν να χρησιμοποιηθούν για εξαγωγή πολύτιμων πληροφοριών και γνώσης από μαζικούς όγκους δεδομένων"*

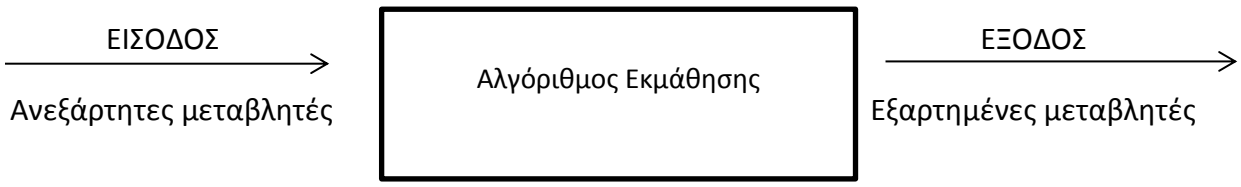
Στις μέρες μας η σημαντικότητα των εφαρμογών του data mining είναι πλέον φανερή σε πολλούς τομείς όπως η βιοστατική, μετεωρολογία, επιχειρήσεις, μηχανική και φυσικά τα χρηματοοικονομικά.

Κλείνοντας την εισαγωγή στο data mining θα αναφέρω ένα απόφθεγμα των Grey και Watson που συνοψίζει πλήρως την χρησιμότητα του data mining*: "Το data mining επιτρέπει σε αναλυτές και διευθυντές καταστημάτων να βρουν τις απαντήσεις χωρίς να έχουν καν ρωτήσει".*

1.2 Τεχνικές εξόρυξης γνώσης

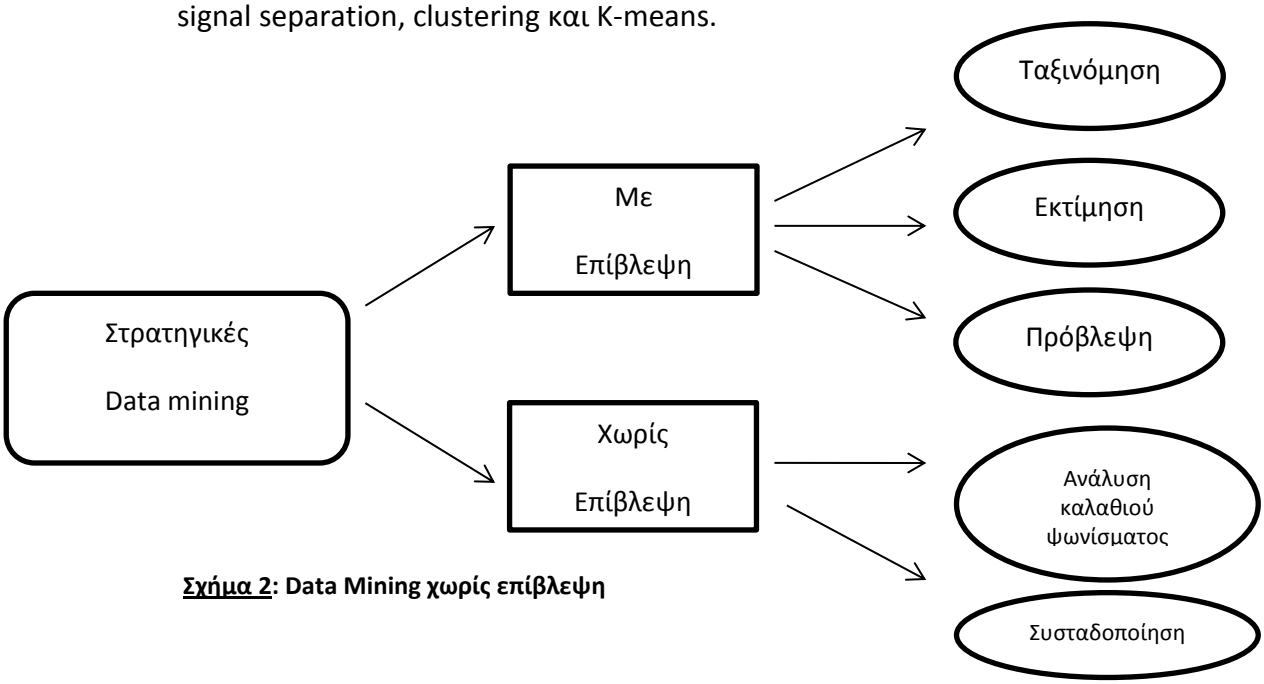
Στο data mining έχουμε 2 βασικές τεχνικές εξόρυξης γνώσης:

- **Μέθοδοι με επίβλεψη (Supervised Methods)** : Εδώ τα δεδομένα μου αποτελούνται από ένα διάνυσμα επεξηγηματικών μεταβλητών (*x*) και μία τιμή απόκρισης (y). Το σκεπτικό πίσω από τις μεθόδους με επίβλεψη είναι με την ανάλυση των δεδομένων μου να δημιουργήσει η μέθοδος μία συνάρτηση ταξινόμησης (ταξινομητής) για να χαρτογραφήσει νέα δεδομένα που δεν συμπεριλαμβάνονται στο αρχικό μου σετ δεδομένων. Μερικά παραδείγματα αυτών των μοντέλων είναι τα νευρωνικά δίχτυα, δέντρα αποφάσεων, λογιστική παλινδρόμηση, μηχανές διανυσματικής υποστήριξης (SVM) , boosting και bagging μέθοδοι.



<u>Σχήμα 1</u>: Data Mining με επίβλεψη

- **Μέθοδοι χωρίς επίβλεψη (Unsupervised Methods):** Αντίθετα με τη μέθοδο εκμάθησης με επίβλεψη αυτή η μέθοδος προσπαθεί να βρει κρυμμένες δομές σε δεδομένα που δεν έχουν μεταβλητή απόκρισης. Άρα δεν έχουμε πρόβλεψη μελλοντικών αποτελεσμάτων αλλά εξερεύνηση των δομών και των σχέσεων των δεδομένων μου. Παραδείγματα αυτών των δομών είναι τα Kohonen Networks, blind signal separation, clustering και K-means.



<u>Σχήμα 2</u>: Data Mining χωρίς επίβλεψη

Μέθοδοι με επίβλεψη:

Ταξινόμηση: Η ταξινόμηση είναι η δημοφιλέστερη και πιο κατανοητή στρατηγική του data mining και στηρίζεται σε 4 βασικά συστατικά:

- **Κλάσεις :** Είναι η εξαρτημένη κατηγορική μεταβλητή του μοντέλου.
- **Ανεξάρτητες μεταβλητές (predictors):** Δίνουν τα χαρακτηριστικά των δεδομένων που συμμετέχουν στη διαδικασία ταξινόμησης.
- **Σετ δεδομένων εκμάθησης (training set):** Το σετ δεδομένων εκμάθησης περιλαμβάνει τα 2 πρώτα συστατικά (κλάσεις και ανεξάρτητες μεταβλητές). Το μοντέλο εκπαιδεύεται σε αυτό το σετ για να προβλέψει τα μελλοντικά σημεία.
- **Σετ δοκιμής:** Νέα δεδομένα στα οποία ελέγχεται η ακρίβεια του μοντέλου μας.

Πιο συγκεκριμένα η ταξινόμηση έχοντας ένα σετ εκμάθησης $D=\{\overrightarrow{x_1},\overrightarrow{x_2},\overrightarrow{x_{13}},.....\overrightarrow{x_n}\}$ και ένα σετ κλάσεων $C=\{C_1, C_2, C_3, \ldots . C_m\}$ προσπαθεί να βρει την βέλτιστη συνάρτηση $f : D \longrightarrow C$.

Παραδείγματα: εύρεση περιοχών επιρρεπή σε σεισμό, να ταξινομήσεις κάποιο ως πετυχημένο, να δώσεις τιμές 0 ή 1 στα στοιχεία σου.

Εκτίμηση: Ο στόχος εδώ είναι να εκτιμήσουμε την τιμή εξόδου μιας άγνωστης μεταβλητής. Η διαφορά αυτής της στρατηγικής με την ταξινόμηση είναι ότι ενώ στην ταξινόμηση η μεταβλητή μας είναι κατηγορική εδώ είναι αριθμητική.

Παραδείγματα: Πιθανότητα κάποιος να πάθει καρδιακή προσβολή, πιθανότητα κάποια μετοχή να ανεβεί, πόσα ξοδεύει κάποιος γνωρίζοντας τον μισθό του.

Πρόβλεψη: Ενώ οι δύο πρώτες στρατηγικές χρησιμοποιούνταν για να εκτιμήσουν συμπεριφορά (παρούσα) η πρόβλεψη χρησιμοποιείται για να προβλέψει τη μελλοντική συμπεριφορά.

Παραδείγματα: Να διευκρινίσει αν κάποιος συνδρομητής μια τηλεφωνικής εταιρείας θα αλλάξει παροχέα στον επόμενο μήνα.

Μέθοδοι χωρίς επίβλεψη

- **Συσταδοποίηση:** Η μέθοδος διαχωρισμού ενός σετ δεδομένων σε διάφορα μέρη (συστάδες). Η δομή γνώσης του προγράμματος αποκτάται με αναφορά στις μετρικές ποιότητας της συστάδας. Στην αρχή της διαδικασίας ο αριθμός των συστάδων πρέπει να είναι γνωστός.
- **Ανάλυση καλαθιού ψωνίσματος (market basket analysis):** Εδώ ανακαλύπτουμε μη διαισθητικές σχέσεις μεταξύ των προϊόντων που πωλούνται. Μετά την ανάλυση ο παροχές μπορεί να πάρει καλύτερες αποφάσεις για τη διαφήμιση, παρουσίαση προϊόντων και με τη σειρά θα βάλει τα προϊόντα του στα ράφια.

1.3 Διαδικασία του Data Mining

Η Ανακάλυψη Γνώσης στα Δεδομένα (Knowledge discovery in databases -KDD) είναι μια αυτοματοποιημένη διαδικασία ανάλυσης και μοντελοποίησης τεράστιων αποθηκών δεδομένων. Αν και υπάρχουν και άλλες διαδικασίες data mining, η KDD είναι η μέθοδος που χρησιμοποιούν κατά κόρον οι data miners. Η KDD χωρίζεται σε 5 στάδια.

1. **Επιλογή δεδομένων:** Σε αυτό το στάδιο παίρνουμε ολόκληρο το δείγμα των δεδομένων και προσπαθούμε να αποκτήσουμε μία βασική ιδέα για το τι μας λένε, αναλύουν τα δεδομένα μας. Επίσης, στο παρών στάδιο θέτουμε τους στόχους και τις προσδοκίες μας.
2. **Προ-επεξεργασία:** Η πολυεπεξεργασία είναι ένα πού σημαντικό στάδιο γα τη σωστή εξόρυξη δεδομένων. Εδώ προετοιμάζουμε το στοχευμένο σύνολο δεδομένων (σετ εκπαίδευσης) από το οποίο και τελικά θα εξάγουμε τη ζητούμενη συνάρτηση. Το σετ εκπαίδευσης πρέπει να είναι αρκετά μεγάλο έτσι ώστε να συμπεριλαμβάνει τις σχέσεις και τους συσχετισμούς του συνόλου των δεδομένων μας αλλά και αρκετά μικρό έτσι ώστε να εξάγει τα δεδομένα σε ένα λογικό πλαίσιο.

3. **<u>Μετασχηματισμός δεδομένων</u>**: Εδώ τα δεδομένα μου “καθαρίζονται”, δηλαδή απομακρύνεται από αυτά ο θόρυβος, αντιμετωπίζονται οι ελλείπουσες τιμές και τέλος μειώνεται το πλήθος τους αν αυτό είναι δυνατό.

4. **<u>Εξόρυξη δεδομένων.</u>** Η Εξόρυξη δεδομένων χωρίζεται σε 6 υποκατηγορίες:

    a. **Εύρεση ανωμαλιών:** Ο προκαθορισμός και η χρήση παράξενων δεδομένων που είτε έχουν κάποιο ιδιαίτερο ενδιαφέρον ή είναι λάθη που πρέπει να διερευνηθούν περεταίρω.
    b. **Σχέση κανόνων εκμάθησης:** Εδώ ανακαλύπτουμε ενδιαφέρων σχέσεις μεταξύ των μεταβλητών μας.
    c. **Ομαδοποίηση:** Ομαδοποιούμε τα δεδομένα σε κατηγορίες με κοινά στοιχεία.
    d. **Ταξινόμηση:** Προσδιορίζουμε σε ποια κατηγορία ανήκουν οι παρατηρήσεις μας
    e. **Παλινδρόμηση :**Προσπαθούμε να βρούμε τη ζητούμενη συνάρτηση που θα κατηγοριοποιεί τα νέα δεδομένα που θα μας παρουσιάσουν
    f. **Σύνοψη:** Παίρνουμε μια πιο συμπαγή σύνοψη των δεδομένων μας, συμπεριλαμβανομένων και αναφορών.

5. **<u>Επεξήγηση/ Εκτίμηση αποτελεσμάτων</u>:** Σε αυτό το στάδιο επαληθεύουμε ότι τα μοτίβα που εξάγαμε από το σετ εκπαίδευσης μας είναι σωστά. . Δεν είναι απαραίτητο ότι όλα τα μοτίβα που έχουμε εξάγει είναι ορθά. Ένα από τα προβλήματα που μπορούν να παρουσιαστούν είναι το overfitting (το μοντέλο που θα επεξηγεί το τυχαίο λάθος (θόρυβο) αντί τη ζητούμενη σχέση). Για να ελέγξουμε την ορθότητα της συνάρτησης που έχουμε εξάγει την δοκιμάζουμε σε ένα σετ επικύρωσης δεδομένων (validation set) που είναι διαφορετικό από το σετ εκπαίδευσης(training set).

1.4 Εφαρμογές

Όπως αναφέραμε και πριν το data mining έχει μια μεγάλη ποικιλία εφαρμογών καθώς είναι ιδανικό εργαλείο για την αντιμετώπιση μεγάλων σετ δεδομένων και λήψη αποφάσεων.

Στις μέρες μας το data mining χρησιμοποιείται κυρίως στην ιατρική, τηλεπικοινωνίες μετεωρολογία, διαφήμιση και φυσικά στα χρηματοοικονομικά των οποίων εφαρμογές θα μελετήσουμε στη συνέχεια .

Εφαρμογές στα Χρηματοοικονομικά.

Σε αντίθεση με τους άλλους κλάδους που εφαρμόζεται το data mining οι εφαρμογές του στα χρηματοοικονομικά υπερτερούν λόγω της ευκολίας συσσώρευσης και καταγραφής δεδομένων σε αντίθεση με άλλους κλάδους όπως η ιατρική και η διαφήμιση, όπου η συλλογή δεδομένων δεν είναι μόνο χρονοβόρα αλλά και πολυέξοδη (μερικές φορές ακόμα και της τάξης των εκατομμυρίων). Οι σημαντικότεροι κλάδοι των χρηματοοικονομικών στους οποίους εφαρμόζεται εκτενώς το data mining είναι η διαχείριση ρίσκου, διαχείριση χαρτοφύλακα, trading, η ταυτοποίηση πελατών και η διαχείριση σχέσεων μεταξύ των πελατών.

- **Διαχείριση χαρτοφύλακα:** Η διαχείριση ρίσκου στους χαρτοφύλακες προσεγγίζει συγκεντρωτικά την ποσοτικοποίηση του ρίσκου ενός συνόλου επενδύσεων. Ο προφανής στόχος σε αυτή την περίπτωση είναι η μεγιστοποίηση των κερδών του χαρτοφύλακα μας καθώς και η ελαχιστοποίηση του ρίσκου. Δηλαδή η μεγιστοποίηση του παράγοντα επιστροφή χρημάτων / ρίσκο. Με τη χρήση του data mining είμαστε σε θέση να αναλύσουμε και να εξάγουμε πολλές επενδυτικές στρατηγικές, μέσω των οποίων τα κεφάλαια μας θα μετακινηθούν σε ένα συνδυασμό επενδύσεων που ανάλογα με την τάση της αγοράς θα μας επιφέρει το μεγαλύτερο κέρδος με το μικρότερο δυνατό ρίσκο. Αυτό πετυχαίνεται με τη ανάλυση των τάσεων της αγοράς, ανάλυση κερδών και απωλειών, συναλλαγματικά έξοδα και επιτόκια.

- **Διαχείριση ρίσκου:** Η διαχείριση και μέτρηση ρίσκου βρίσκονται στον πυρήνα κάθε χρηματοπιστωτικού ιδρύματος. Η διαχείριση ρίσκου χωρίζεται σε 2 μεγάλες υποκατηγορίες (ρίσκο χρηματοπιστωτικής αγοράς και πιστωτικό ρίσκο) οι οποίες υπέστησαν μεγάλες αλλαγές βασισμένες στις εξειδικευμένες μεθόδους data mining.
    - **Ρίσκο χρηματοπιστωτικής αγοράς:** Για τα όργανα του χρηματοπιστωτικού ιδρύματος όπως οι χρηματιστηριακοί δείκτες, επιτόκια και συνάλλαγμα η μέτρηση του ρίσκου βασίζεται σε κάποιους παράγοντες ρίσκου όπως τόκοι και οικονομική ανάπτυξη. Το κύριο ενδιαφέρον έρευνας και εφαρμογών βρίσκεται στην συναρτησιακή μορφή μεταξύ των οργάνων του χρηματοπιστωτικού συστήματος και στους παράγοντες ρίσκου. Υπάρχουν πολλές διαφορετικές προσεγγίσεις της μέτρησης του ρίσκου, όλες εξ αυτών βασισμένες σε μοντέλα που αντιπροσωπεύουν την αλληλεξάρτηση των παραγόντων με την συνολική αγορά. Οι περισσότερες από αυτές τις μεθόδους μπορούν να κατασκευαστούν μόνο με τη χρησιμοποίηση τεχνικών data mining σε ιδιωτικά δεδομένα που χρειάζονται συνεχή επίβλεψη.

    - **Πιστωτικό Ρίσκο:** Η ανάλυση του πιστωτικού ρίσκου είναι το κύριο συστατικό στη διαδικασία των εμπορικών δανείων. Χωρίς αυτό ο εκάστοτε δανειστής δεν θα είχε κανένα στοιχείο για το ρίσκο του δανεισμού και άρα θα ήταν ανίκανος να αποφασίσει για το αν θα δανείσει τα χρήματά του και με τη όρους θα τα δανείσει. Εδώ βασικά το ρίσκο αντιπροσωπεύει την πιθανότητα μη επιστροφής των χρημάτων στον δανειστή. Σε αυτή την περίπτωση το μεγαλύτερο μέρος του συστήματος διαχείρισης ρίσκου θα είναι ένα τυπικό πρόβλημα data mining με επεξηγηματικές μεταβλητές την «αξία» του πιστωτή, ρυθμό επιστροφής χρημάτων, προηγούμενα δάνεια κτλ.

- **Trading:** Ένας από του περισσότερο αναπτυσσόμενους τομείς του data mining είναι η προσπάθεια για κτίσιμο εργαλείων trading, που βασισμένα στα ιστορικά δεδομένα της αγοράς θα είναι σε θέση να προβλέψει τη βραχυπρόθεσμη κίνηση των επιτοκίων, συναλλαγμάτων και μετοχών. Ο στόχος μας είναι να καταφέρουμε να εντοπίσουμε πότε η αγορά είναι “φτηνή” και πότε είναι “ακριβή”. Δηλαδή να εξάγουμε συμβουλές για το πότε η μια μετοχή/συνάλλαγμα είναι υπερεκτιμημένη ή υποτιμημένη με στόχο την πώληση/αγορά του ανάλογου αγαθού τη συγκεκριμένη χρονική στιγμή.

  Αν και παραδοσιακά οι αγοραπωλησίες βασίζονται στο ένστικτο και πείρα των trader ακόμα και ο πιο έμπειρος trader μπορεί να συνυπολογίσει ένα μικρό αριθμό παραγόντων καθιστώντας την απόφασή του αμφιλεγόμενη. Λόγω των πολλών δεδομένων που είναι διαθέσιμα για την κίνηση των μετοχών/συναλλάγματος κτλ είναι προφανές πως το data mining είναι ιδανική επιλογή, αν όχι για πλήρη λήψη αποφάσεων, για βοηθητικό εργαλείο για κάθε trader έτσι ώστε να βρίσκει μοτίβα και δομές που ο ίδιος θα ήταν αδύνατο να ανακαλύψει.

- **Ξέπλυμα βρώμικου χρήματος:** Η αστυνομική λογιστική είναι ένας τομέας υπεύθυνος για την ανακάλυψη παράνομων οικονομικών συναλλαγών με κυριότερο σκοπό την εύρεση επιχορηγήσεων μηχανισμών τρομοκρατίας, όπου “καθαρά ” και “βρώμικα” κεφάλαια χρησιμοποιούνται για αγορά και κατασκευή όπλων. Ο στόχος του data mining σε αυτόν τον τομέα είναι ο εντοπισμός ύποπτων συναλλαγών καθώς και η μείωση των «λανθασμένα θετικών» ύποπτων συναλλαγών. Αυτό μπορεί να επιτευχθεί με το data mining ψάχνοντας για ύποπτους πολύπλοκους συνδυασμούς συναλλαγών με χρησιμοποίηση κυρίως μεθόδων χωρίς επίβλεψη.

Κεφάλαιο 2: Τεχνικές εξόρυξης γνώσης

2.1 Τεχνητά Νευρωνικά δίχτυα

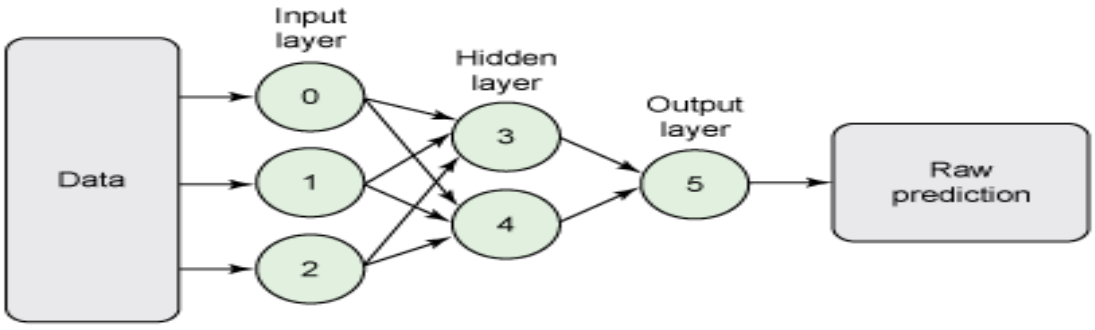
2.1.1. Εισαγωγή

Ορισμός: Ένα Τεχνητό Νευρωνικό Δίχτυο (Artificial Neural Network-ANN) είναι ένα υπολογιστικό μοντέλο βασισμένο στη δομή και τις συσχετίσεις των βιολογικών νευρωνικών διχτύων. Οι πληροφορίες που ρέουν μέσα από το δίχτυο επηρεάζουν τη δομή του, δηλαδή κατά μία έννοια το νευρωνικό δίχτυο “μαθαίνει” βάση των δεδομένων εισόδου- εξόδου.

Τα ANN είναι ένα πολύ χρήσιμο και ευέλικτο στατιστικό εργαλείο λόγω της προσαρμοστικότητάς τους. Θεωρούνται ως μη γραμμικά στατιστικά μοντέλα ανάλυσης δεδομένων όπου οι πολύπλοκες σχέσεις μεταξύ εισόδου και εξόδου μοντελοποιούνται και ανακαλύπτονται.

Τα σημαντικότερα πλεονεκτήματα των νευρωνικών δικτύων που πρέπει να τονίσουμε είναι η ικανότητά τους να «εκπαιδεύονται» παρακολουθώντας και αναλύοντας τα δεδομένα μας. Αυτά τα εργαλεία μας βοηθούν να υπολογίσουμε τη βέλτιστη μέθοδο λύσης καθώς ορίζουν υπολογιστικές συναρτήσεις και κατανομές βασισμένα μόνο σε ένα μέρος των δεδομένων μας παρά στο σύνολό τους

Τα ANN αποτελούνται από τρία αλληλένδετα επίπεδα. Το πρώτο αποτελείται από την είσοδο των νευρώνων όπου και εισάγουμε τα δεδομένα μας (INPUT), τα οποία στέλνουν δεδομένα στο $2^{ο}$ επίπεδο (hidden layer) για ανάλυση και με τη σειρά τους στέλνουν τη μεταβλητή εξόδου στο $3^{ο}$ επίπεδο (output) που μας δίνει την πρόβλεψη (raw prediction).



Σχήμα 3: Σχηματικά ένα νευρωνικό δίκτυο

2.1.2 Βιολογικά νευρωνικά δίχτυα.

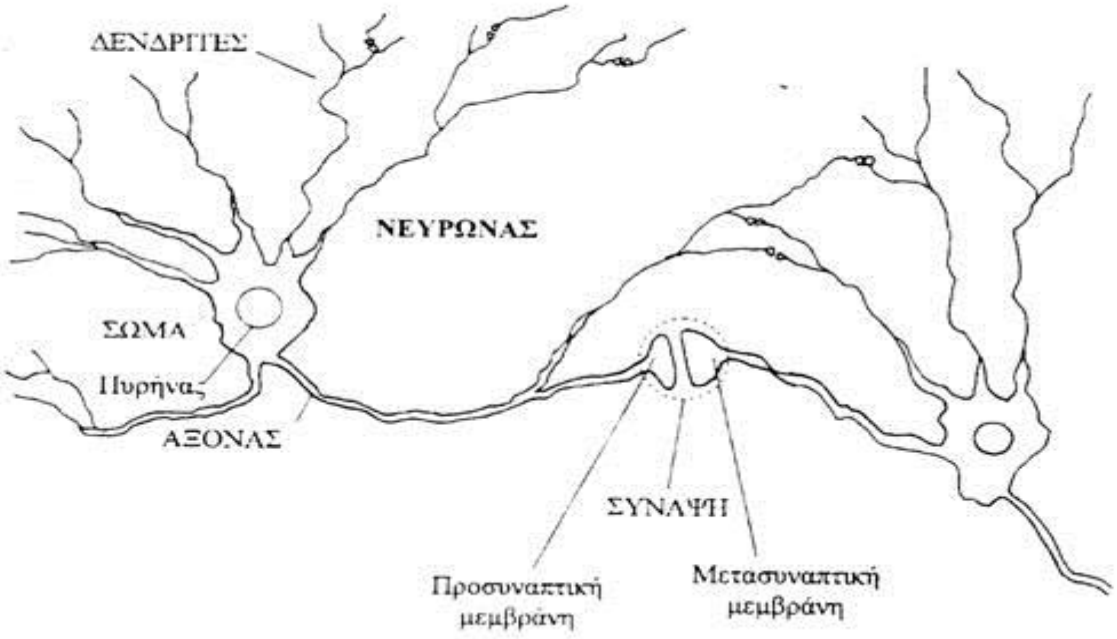
Για να κατανοήσουμε τα τεχνητά νευρωνικά δίχτυα είναι σημαντικό πρώτα να καταλάβουμε πως λειτουργούν τα βιολογικά νευρωνικά δίχτυα από τα οποία και εμπνεύστηκαν τα ANN.

Κατά τη διάρκεια των χρόνων, την προσπάθειά μας να εξηγήσουμε το πώς λειτούργει ο ανθρώπινος εγκέφαλος και πως μεταφέρονται οι πληροφορίες μέσα στον εγκέφαλο καταλήξαμε στο (αν και απλουστευμένο, περιεκτικό) συμπέρασμα πως συμπεριφέρεται σαν έναν τεραστίων διαστάσεων, μη γραμμικός υπολογιστής. Ο εγκέφαλος έχει την μοναδική ικανότητα να οργανώνει τους νευρώνες (από τους οποίους και αποτελείται) με τέτοιο τρόπο ώστε να κάνει απίστευτα πολύπλοκες διαδικασίες και υπολογισμούς όπως σκέψη, κίνηση, υπολογισμούς και αντίληψη σε μεγάλες ταχύτητες (της τάξης των millisecond).

Οι νευρώνες είναι η βασική χαρακτηριστική μονάδα του νευρικού μας συστήματος, που έχει ως κύριο ρόλο την επικοινωνία. Οι νευρώνες λαμβάνουν πληροφορίες (δεδομένα) από τα κύτταρα και την μεταφέρουν σε άλλα κύτταρα. Αποτελούνται από 4 βασικά χαρακτηριστικά.

- **Πυρήνα:** Αποτελείται από νεκλόνια και είναι υπεύθυνος για τις διεργασίες κάθε νευρώνα.
- **Δενδρίτες:** Προεκτάσεις που είναι υπεύθυνες για την εισροή πληροφοριών στον πυρήνα.
- **Άξονες:** Μεταφέρουν πληροφορίες μακριά από τον πυρήνα.
- **Τερματικός Άξονας:** Το σημείο από το οποίο η πληροφορία εξέρχεται από το νευρώνα. Οι τερματικοί άξονες σχηματίζουν συνάψεις με δενδρίτες των γειτονικών νευρώνων.

Οι βιολογικοί νευρώνες ενεργοποιούνται με χημικά που είναι παγιδευμένα στις συνάψεις. Μόλις ενεργοποιηθεί ο νευρώνας η πληροφορία μεταφέρεται μέσα του μέσω ενός ηλεκτρικού παλμού. Μόλις ο ηλεκτρικός παλμός φτάσει από τους δενδρίτες στον τερματικό άξονα εκπέμπει τα κατάλληλα χημικά στοιχεία στους γειτονικούς νευρώνες μεταφέροντας την πληροφορία .



<u>Εικόνα 1</u>: Δύο βιολογικοί νευρώνες

2.1.3. Μαθηματικό μοντέλο

Τα τεχνητά νευρωνικά δίχτυα απέχουν πολύ από τις ικανότητες των βιολογικών νευρωνικών διχτύων λόγω του ότι οι νευρώνες στα τεχνητά νευρωνικά δίχτυα είναι της τάξεως των εκατοντάδων (το πολύ χιλιάδων) σε αντίθεση με αυτούς του εγκεφάλου που είναι της τάξεως των δισεκατομμυρίων.

Πάμε τώρα να αναλύσουμε το μαθηματικό μοντέλο των ANN, θα ξεκινήσουμε με μία μακροσκοπική ματιά στο perceptron πολλών στρωμάτων (MultiLayer Perceptron-MLP) για να δούμε το σκεπτικό και τους στόχους του, στη συνέχεια θα δούμε αναλυτικά τι συμβαίνει σε κάθε νευρώνα και μετά θα αναλύσουμε τον perceptron ενός στρώματος με μία εφαρμογή και θα κάνουμε γενίκευση στον MLP.

Έχουμε ένα πρόβλημα n διαστάσεων και c κλάσεων. Στην είσοδο του ANN έχουμε ένα διάνυσμα **x =** $\{ x_1, x_2, \ldots . x_n\}$ **∈** $\mathbb{R}^{\boldsymbol{n}}$ **,** το οποίο παράγει τιμές για τις c διακριτές συναρτήσεις $\{g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots \ldots, g_c(\boldsymbol{x})\}$ στην έξοδο του. Το ANN προσπαθεί να ελαχιστοποιήσει το τετραγωνικό σφάλμα σε ένα σετ εκπαίδευσης του οποίου οι τιμές είναι γνωστές. Θεωρούμε το σετ εκπαίδευσης ως εξής:

$$Z = \{ \boldsymbol{z}_1 , \boldsymbol{z}_2 , \ldots \ldots \boldsymbol{z}_n\} \mu\varepsilon \ \boldsymbol{z}_j \ \epsilon \ R^n \ \gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ j = 1 \ldots . n \qquad (2.1.3.0.1)$$

Και $l(\boldsymbol{z}_j) \epsilon \ \Omega$ *δηλαδή ανήκει στον δειγματικό μου χώρο. Η συνάρτηση που πρέπει να ελαχιστοποιηθεί είναι η:*

$$E = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{c}\{ g_i(\boldsymbol{z_j}) - \ I\big(\omega_\iota, l\big(\boldsymbol{z_j}\big) \ \big\}^2 \qquad (2.1.3.0.2)$$

Όπου το $I(\omega_\iota, l\big(\boldsymbol{z_j}\big))$ είναι η δείκτρια συνάρτηση που παίρνει τιμή 1 αν $l\big(\boldsymbol{z_j}\big)=\omega_\iota$ και 0 αλλιώς. Έχει δειχθεί ότι το

$$\lim_{n\to\infty} g_i(\boldsymbol{x}) = P(\omega_\iota \ / \ \boldsymbol{x}) \ \mu\varepsilon \ \boldsymbol{x} \ \epsilon \ R^n$$

που είναι και το ζητούμενο. Έχει αποδειχθεί ότι η προσεγγιστική θεωρία ισχύει για τα MLP (για την απόδειξη παραπέμπουμε στη βιβλιογραφία [3] και [29])

2.1.3.1 Νευρώνες

Πάμε τώρα να δούμε τη γίνεται στους νευρώνες του ANN.

Έστω **u =** $\{ u_0, u_1, \ldots . u_q\}$ **∈** $\mathbb{R}^{q+1}$ και v **∈** $\boldsymbol{R}$ η έξοδος του. Χρησιμοποιούμε συναπτικά βάρη **w =** $\{ w_0, w_1, \ldots . w_q\}$ **∈** $\mathbb{R}^{q+1}$ **.** Ο νευρώνας υλοποιεί τα :

$$v = \varphi(\xi) \quad \mu\varepsilon \quad \xi = \sum_{i=0}^{q} w_i u_i$$

Με ***φ: R->R*** συμβολίζεται η συνάρτηση ενεργοποίησης. Η συνάρτηση ενεργοποίησης έχει ως σκοπό να κρατήσει τις εξόδους των νευρώνων σε ένα συγκεκριμένο όριο (συνήθως 0 και 1 ή 1 και -1). Έχουμε 3 πιθανές συναρτήσεις ενεργοποίησης φ(.).

Σε αυτό το στάδιο θα προσθέσουμε και τον παράγοντα της μεροληψίας b. Σκοπός της μεροληψίας είναι η αύξηση ή η μείωση της τιμής που δίνει το δίκτυο στη συνάρτηση ενεργοποίησης ανάλογα με το αν είναι αρνητικό ή θετικό. Η μεροληψία είναι το βάρος $-w_0$, άρα έχουμε

$$v = \varphi(\zeta - (-w_0)) = \varphi(\sum_{i=1}^{q} w_i u_i - (-w_0))$$

Στη συνέχεια θέτω $\xi = \sum_{i=0}^{q} w_i u_i - (-w_0)$

Συνάρτηση ενεργοποίησης:

- **Συνάρτηση κατωφλιού (threshold)**

$$\Phi(\xi) = \begin{cases} 1 \ \alpha\nu \ \xi \ \geq \theta \\ 0 \ \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Η συνάρτηση κατωφλιού έχει ως έξοδο το 1 αν το $\xi \geq \theta$ όπου το θ είναι μία τιμή κατώφλι που θέσαμε εμείς και 0 αλλού.

- **Σιγμοειδής συνάρτηση (sigmoid function)**

$$\Phi(\xi)= \frac{1}{1+\exp(-\xi)}$$

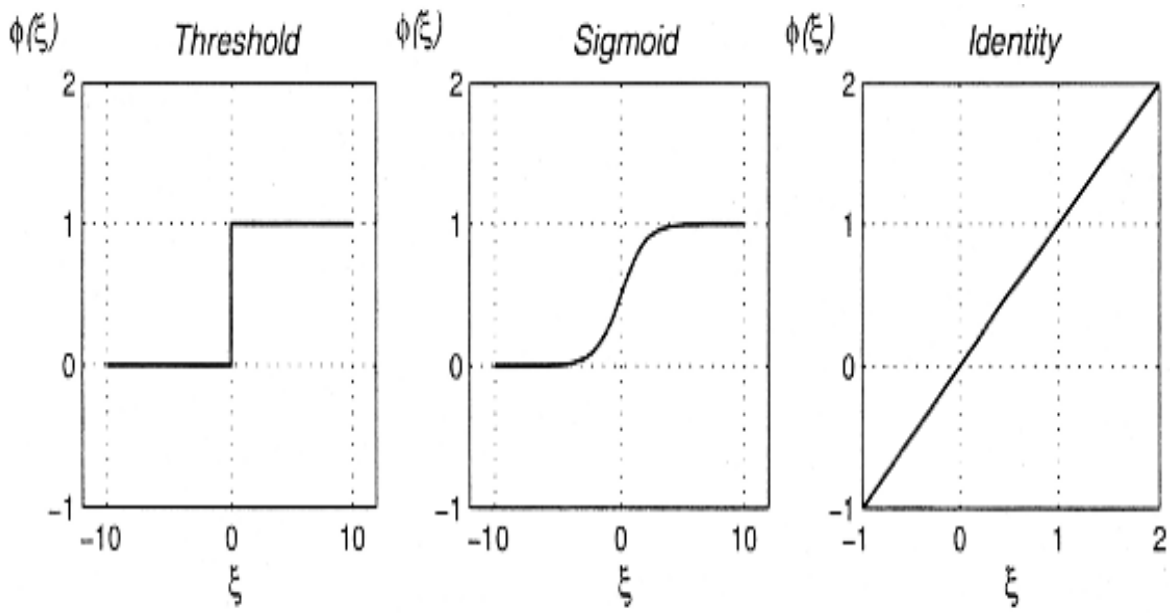
Είναι μια γνησίως αύξουσα διαφορίσιμη συνάρτηση που είναι ομαλή και ασυμπτωτική.

- **Ταυτοτική συνάρτηση (identity) :**

$$\Phi(\xi) = \xi' \text{ για } 0 \leq \xi \leq 1$$

Αν και δεν χρησιμοποιείται εκτενώς η ταυτοτική συνάρτηση σαν συνάρτηση ενεργοποίησης σε συνδυασμό με την συνάρτηση κατώφλι μας δίνει την τμηματική γραμμική συνάρτηση

$$\Phi(\xi) = \begin{cases} 0 \ \alpha\nu \ \xi \leq 0 \\ \xi \ \alpha\nu \ 0 \leq \xi \leq 1 \\ 1 \ \alpha\nu \ \xi \leq 1 \end{cases}$$



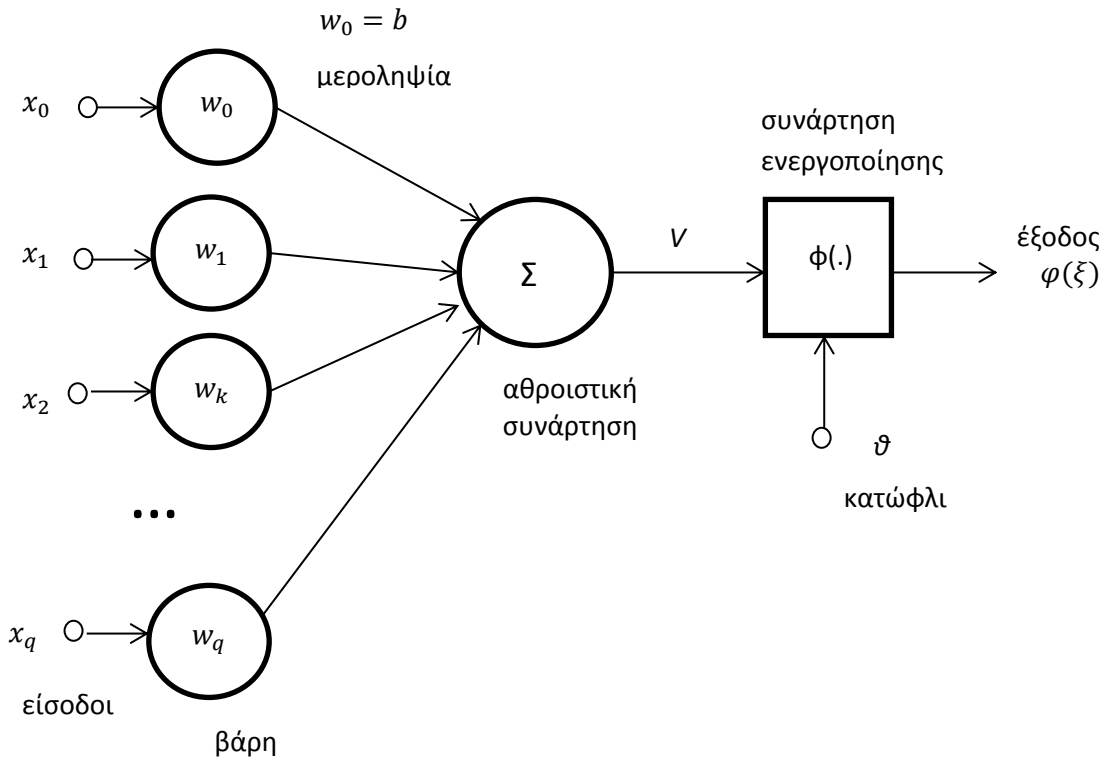
<u>Εικόνα 2</u>: Η συναρτήσεις κατώφλι, σιγμοειδής και ταυτοτική αντίστοιχα

Από τις 3 συναρτήσεις ενεργοποίησης η σιγμοειδής είναι αυτή που χρησιμοποιείται συχνότερα λόγω του ότι:

a) Μοντελοποιεί γραμμικές συναρτήσεις και συναρτήσεις κατωφλιού. Είναι σχεδόν γραμμική στην αρχή ενώ για μεγάλα βάρη είναι πρακτικά η συνάρτηση κατωφλιού

b) Όπως προαναφέραμε είναι διαφορίσιμη με παράγωγο στο ξ την Φ'(ξ)=Φ(ξ)[1-Φ(ξ)].

Επίσης, αξίζει να σημειωθεί πως γεωμετρικά η συνάρτηση
$v = \varphi(\zeta - (-w_0)) = \varphi\left(\sum_{i=1}^{q} w_i u_i - (-w_0)\right)$ είναι ένα υπερεπίπεδο στον R^n.



<u>Σχήμα 4</u>: Σχηματικά ένας νευρώνας

2.1.4 Τοπολογία Νευρωνικών Δικτύων

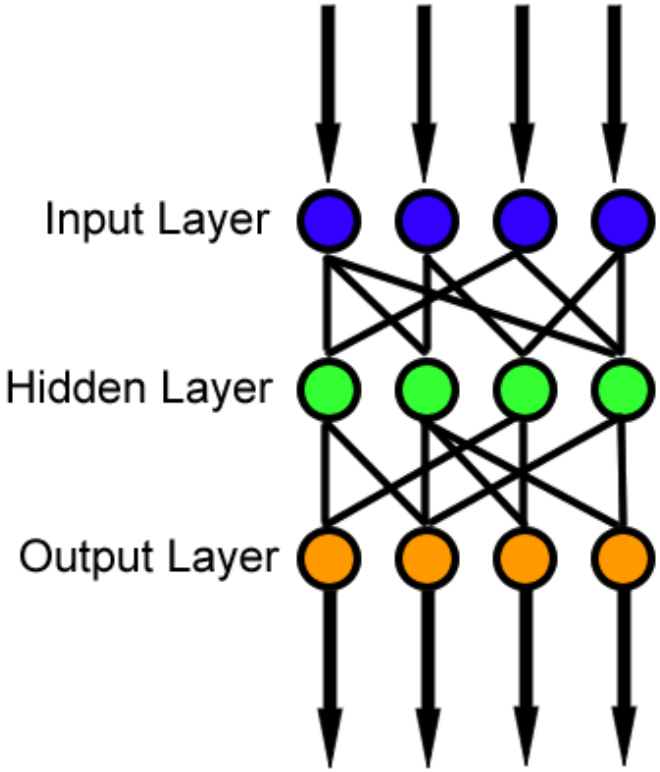
Μέχρι τώρα είδαμε τη δομή και τις ιδιότητες της βασικής μονάδας επεξεργασίας σε ένα τεχνητό νευρωνικό δίκτυο. Παρακάτω θα επικεντρωθούμε σε πρότυπα των συνδέσεων μεταξύ των μονάδων και της διάδοσης των δεδομένων. Μπορούμε να πούμε ότι υπάρχουν δυο διαφορετικές κλάσεις αρχιτεκτονικών δικτύου.

Εμπρόσθιας τροφοδότησης Νευρωνικά Δίκτυα (Feed-forward neural networks)

Τα νευρωνικά δίκτυα εμπρόσθιας τροφοδότησης είναι ο πρώτος και ο πιο απλός τύπος νευρωνικών δικτύων που έχουν σχεδιαστεί. Στα δίκτυα αυτά, η πληροφορία κινείται μόνο προς μία κατεύθυνση, εμπρόσθια, από τους κόμβους εισόδου, διάμεσο κάποιων *κρυφών νευρώνων* (εφόσον υπάρχουν) και καταλήγει στους κόμβους εξόδου. Οι κρυφοί νευρώνες είναι κόμβοι που παρεμβάλλονται μεταξύ των κόμβων εισόδου και εξόδου του δικτύου και χρησιμοποιούνται σε δίκτυα τα οποία απαιτούν την προσέγγιση συναντήσεων μεγάλης πολυπλοκότητας.

Σε δίκτυα εμπρόσθιας τροφοδότησης δεν υπάρχουν κύκλοι ή βρόγχοι. Μπορούμε να εντάξουμε τους κόμβους του δικτύου σε τρία είδη επιπέδων. Στο **επίπεδο εισόδου (input layer)** όπου εισέρχονται τα στοιχεία των προτύπων εισόδου στο δίκτυο με τη μορφή διανυσμάτων. Στη συνέχεια, αυτά γίνονται είσοδοι για τους νευρώνες των **κρυφών επιπέδων (hidden layers),**, όπου οι έξοδοι του επιπέδου εισόδου γίνονται έξοδοι του πρώτου κρυφού επιπέδου, οι έξοδοι του πρώτου κρυφού επιπέδου είσοδοι του δεύτερου κτλ. Η διαδικασία αυτή συνεχίζεται μέχρι να φτάσουμε στο **επίπεδο εξόδου (output layer).** Η έξοδος του επιπέδου αυτού, είναι και η απάντηση του δικτύου για τα δεδομένα που έχουν εισαχθεί στους κόμβους εισόδου.

Παραδείγματα δικτύων εμπρόσθιας τροφοδότησης είναι τα Perceptron που θα δούμε αναλυτικά στη συνέχεια και τα Adaline.

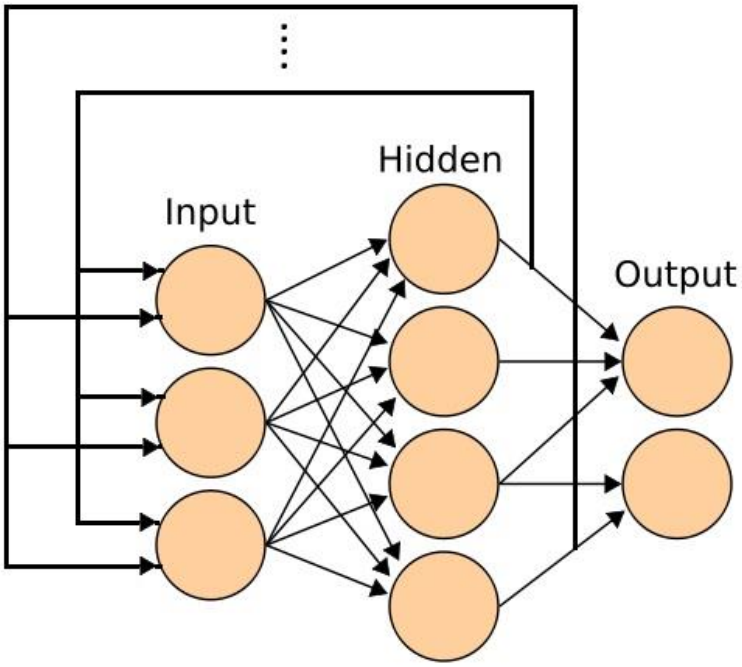


Εικόνα 3: Νευρωνικό Δίκτυο Εμπρόσθιας Τροφοδότησης

Στην εικόνα βλέπουμε ένα παράδειγμα νευρωνικού δικτύου εμπρόσθιας τροφοδότησης με ένα κρυφό επίπεδο. Στο δίκτυο βλέπουμε ότι δεν είναι όλοι οι κόμβοι κάθε επιπέδου συνδεδεμένοι με όλους τους κόμβους του επόμενου επιπέδου για αυτό και λέγεται **μερικώς συνδεδεμένο (partially connected) δίκτυο.** Στα δίκτυα τα οποία υπάρχουν όλες οι συνδέσεις λέγονται **πλήρως συνδεδεμένα (fully connected) δίκτυα.** Επίσης το πιο πάνω δίκτυο αναφέρεται σαν ένα 4-4-4 δίκτυο γιατί έχει 4 κόμβους εισόδου, 4 κόμβους στο κρυφό επίπεδο και 4 κόμβους εξόδου. Γενικά λέμε **m-h1-h2-…..-hn-q δίκτυο,** ένα δίκτυο με m κόμβους εισόδου, h1 κόμβους $1^{ου}$ επιπέδου . . . . . και q κόμβους στο επίπεδο εξόδου.

Αναδραστικά Νευρωνικά Δίκτυα (Recurrent Neural Nettworks RNN)

Στα αναδραστικά νευρωνικά δίκτυα υπάρχουν συνδέσεις μεταξύ των κόμβων, οι οποίες σχηματίζουν κύκλους, δηλαδή σε τουλάχιστον έναν νευρώνα το σήμα εξόδου του επηρεάζει το σήμα που έρχεται στην είσοδο του νευρώνα. Αυτό δημιουργεί μια εσωτερική κατάσταση του δικτύου, που επιτρέπει στο δίκτυο να ανατροφοδοτείται. Σε αντίθεση με τα δίκτυα εμπρόσθιας τροφοδότησης, τα αναδραστικά δίκτυα μπορούν να χρησιμοποιήσουν την εσωτερική τους μνήμη για να επεξεργαστούν αυθαίρετες ακολουθίες εισόδου.



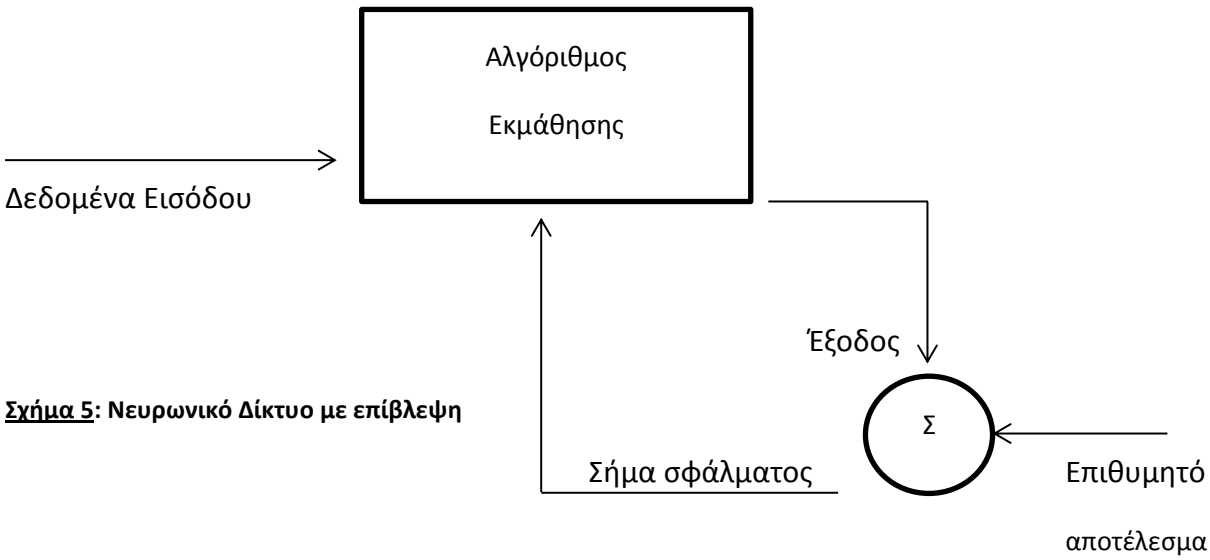
<u>Εικόνα 4</u>: Αναδραστικό Νευρωνικό Δίκτυο

Στα δίκτυα αυτά, οι δυναμικές ιδιότητες είναι σημαντικές. Σε κάποιες περιπτώσεις, οι παράμετροι του δικτύου τροποποιούνται έτσι ώστε το δίκτυο να φτάσει σε μια σταθερή και μόνιμη κατάσταση. Σε άλλες περιπτώσεις όμως, η αλλαγή των παραμέτρων στους νευρώνες εξόδου είναι σημαντική. Έτσι για τα δίκτυα αυτά πολλές φορές χρειάζεται η θεωρία των δυναμικών συστημάτων για να τα αναλύσει.

2.1.5 Ταξινόμηση νευρωνικών αλγορίθμων

Υπάρχουν τρεις μεγάλες κλάσεις νευρωνικών αλγορίθμων. Το πια μέθοδο θα χρησιμοποιήσουμε στο νευρωνικό μας δίχτυο εξαρτάται από το σε τι αποτελέσματα θέλουμε να καταλήξουμε, δηλαδή ο στόχος μας είναι όταν το δίχτυο πάρει ένα σετ εισόδου (input) να παράγει το επιθυμητό σετ εξόδου(output).

- **Μάθηση με επίβλεψη:** Όπως προαναφέραμε στη μάθηση με επίβλεψη έχουμε το ζεύγος (**x,**y) , με το διάνυσμα **x** να είναι η ανεξάρτητες μεταβλητές (μεταβλητές εισόδου) και το y να είναι η εξαρτημένη μεταβλητή απόκρισης. Ο στόχος μας είναι να βρούμε τη βέλτιστη f: X-> Y. Εδώ η συνάρτηση κόστους σχετίζεται με την αντιστοιχία μεταξύ της χαρτογράφησης μας και των πραγματικών δεδομένων, η συνάρτηση κόστους “σιωπηρά” περιέχει προηγούμενη γνώση για το πρόβλημά μας.
  Ο πιο συνήθης τρόπος βελτιστοποίησης της συνάρτησης μας είναι η μέθοδος των ελαχίστων τετραγώνων που προσπαθεί να ελαχιστοποιήσει το τετραγωνικό σφάλμα μεταξύ της f και του y.
  Στα μοντέλα εκμάθησης με επίβλεψη συχνά συμπεριλαμβάνεται η αναγνώριση προτύπων (ταξινόμηση) και η παλινδρόμηση.
  Η μέθοδος με επίβλεψη συχνά επεξηγείται και με τη βοήθεια ενός εξωτερικού παράγοντα, τον “δάσκαλο”, που δεν είναι τίποτα άλλο από μία συνάρτηση που σου δίνει συνεχείς πληροφορίες για την ποιότητα της λύσης που έχεις φτάσει.



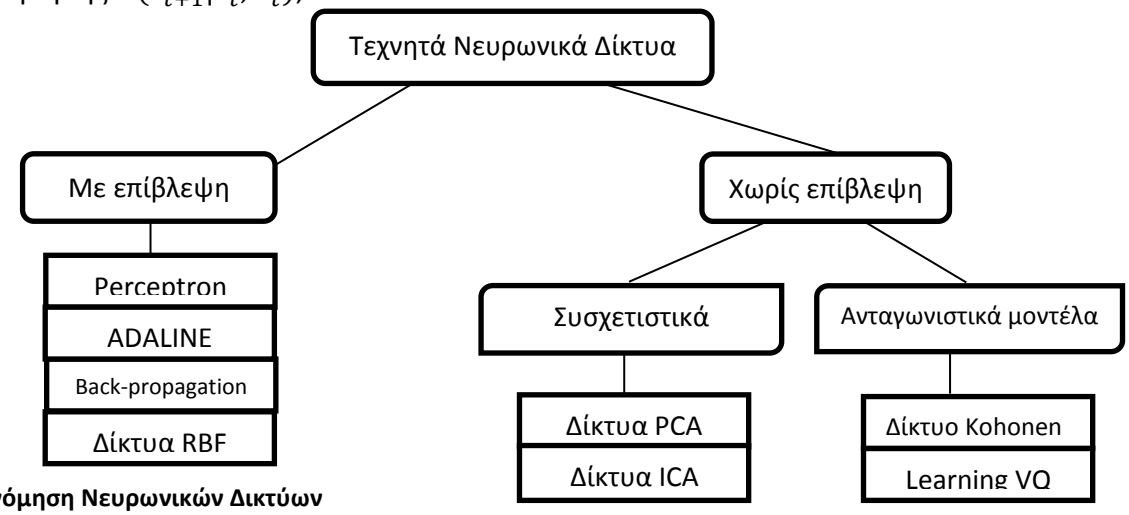
Σχήμα 5: Νευρωνικό Δίκτυο με επίβλεψη

- **Μέθοδος χωρίς επίβλεψη:** Εδώ δίνεται ένα διάνυσμα **x** και στόχος είναι η ελαχιστοποίηση της συνάρτησης κόστους η οποία μπορεί να είναι οποιαδήποτε συνάρτηση των δεδομένων μας. Η συνάρτηση εξόδου εξαρτάται από το τη προσπαθούμε να μοντελοποιήσουμε και τις a priori υποθέσεις μας (τις ιδιότητες του μοντέλου, τις παραμέτρους του και τις παρατηρούμενες μεταβλητές).

  Ένα απλοϊκό μοντέλο είναι το *f(x) = α* με το *α* σταθερά και συνάρτηση κόστους C=E[(x- $f(x))^2$]. Με την ελαχιστοποίηση το C θα μας δώσει α ίσο με το μέσο του μοντέλου μου. Εννοείτε πως η συνάρτηση κόστους μπορεί να είναι πολύ πιο πολύπλοκη.

  Συμπεριλαμβάνει γενικά προβλήματα πρόβλεψης, συσταδοποίηση, εκτίμηση στατιστικών κατανομών, συμπίεση και φιλτράρισμα.

- **Ενισχυτική μάθηση:** Η ενισχυτική μάθηση είναι ένας συνδυασμός των δύο προηγούμενων μορφών. Συγκεκριμένα συνήθως το διάνυσμα δεν δίνεται αλλά αναπαράγεται από ένα συντελεστή που αλληλεπιδρά με το περιβάλλον. Κάθε χρονική στιγμή t ο συντελεστής κάνει μία ενέργεια y_t και το περιβάλλον παράγει μια παρατήρηση x_t και μία στιγμιαία συνάρτηση κόστους C_t. Ο στόχος είναι να ανακαλύψουμε κάποια "πολιτική" για επιλογή ενεργειών που ελαχιστοποιούν κάποια μέτρα Κόστους. Συγκεκριμένα το περιβάλλον μοντελοποιείται ως Μαρκοβιανή αλυσίδα (MDP) με στάδια $s_1\ s_2 \dots s_n\ \epsilon\ S$ και ενέργειες $\alpha_1\ \alpha_2 \dots \alpha_m \epsilon$ A και κατανομή κόστους $P(c_t|s_t)$, κατανομή παρακολούθησης $P(x_t|s_t)$ και μεταπήδησης $P(s_{t+1}|s_t, a_t)$,



Σχήμα 6: Ταξινόμηση Νευρωνικών Δικτύων

2.1.6 Perceptron

Ο Perceptron είναι ένας αλγόριθμος μάθησης με επίβλεψη που επινοήθηκε από τον αμερικάνο ψυχολόγο Φρανκ Ρόσενβαλτ το 1962. Είναι ένας γραμμικός ταξινομητής (κάνει τις προβλέψεις του στηριζόμενος σε γραμμικές συναρτήσεις πρόβλεψης) συνδυάζοντας τα βάρη με τα δοσμένα διανύσματα. Είναι βασικά απλοποιημένες μορφές του βιολογικού νευρικού συστήματος. Θα ξεκινήσουμε την ανάλυση με το perceptron ενός στρώματος.

2.1.6.1 Perceptron ενός στρώματος

Καταρχάς ο perceptron ενός στρώματος εφαρμόζει μια συνάρτηση κατωφλιού

$$\Phi(\xi)=\begin{cases}1 \ \alpha\nu \ \xi \ \geq 0 \\ -1 \ \alpha\lambda\lambda\iota\acute{\omega}\varsigma\end{cases} \qquad \xi = \sum_{i=0}^{q} w_i u_i - (-w_0)$$

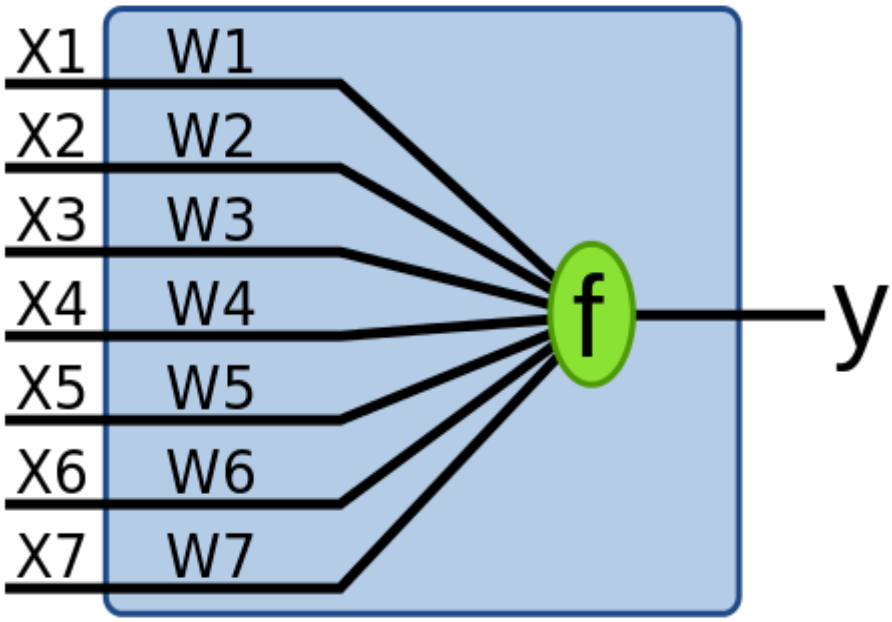
με ξ=**wx** + b, με **w** το βάρος και **x** το διάνυσμα.

Αυτός ο ταξινομητής ενός νευρώνα χωρίζει δύο κλάσεις στον $\boldsymbol{R}^n$ με ένα γραμμικό υπερεπίπεδο $\xi = \sum_{i=0}^{q} w_i u_i - (-w_0)$=0. Ο αλγόριθμος ξεκινά με αρχικά βάρη **w** και τα αλλάζει καθώς κάθε διάνυσμα από το σετ εκπαίδευσης Z (2.1.3.0.1) υποβάλλεται στην είσοδο του perceptron. Η αλλαγή λαμβάνει χώρα μόνο αν στο διάνυσμα $\boldsymbol{z_j}$ έχει δοθεί λάθος κλάση (είναι δηλαδή στη λάθος πλευρά του υπερεπιπέδου). Τα ανανεωμένα βάρη υπολογίζονται:

w* <- *w* - *vηz*$_j$** όπου ***v=f($\boldsymbol{z_j}$) και η παράμετρος ***η*** που καθορίζει την ταχύτητα εκμάθησης του αλγορίθμου. Δύο σημαντικά στοιχεία της εκπαίδευσης του perceptron είναι:

1. Αν δύο κλάσεις είναι γραμμικά διαχωρίσιμες στο R^n (από 1 υπερεπίπεδο) ο αλγόριθμος συγκλίνει πάντα, σε πεπερασμένο αριθμό επαναλήψεων, σε μια γραμμική διάκριση συνόρων χωρίς σφάλμα στο Z για κάθε ***η*** (θεώρημα σύγκλησης perceptron).
2. Αν δύο κλάσεις δεν είναι γραμμικά διαχωρίσιμες στο R^n, ο αλγόριθμος θα επαναλαμβάνεται επ' άπειρο στο Z και ποτέ δεν θα συγκλίνει. Επίσης δεν υπάρχει καμία εγγύηση πως τη στιγμή που θα τερματίσουμε τον αλγόριθμο, η συνάρτηση που θα πάρουμε θα είναι η βέλτιστη.

Για αυτό το λόγο συνήθως τα 10000 βήματα ορίζονται ως το όριο των επαναλήψεων. Αν φτάσουμε αυτό τον αριθμό και υπάρχουν ακόμα αντικείμενα τα οποία είναι ταξινομημένα λάθος ο αλγόριθμος σταματά και δηλώνει πως οι κλάσεις δεν είναι διαχωρίσιμες.



Σχήμα 7: Απλό Perceptron

2.1.6.1.1. Αλγόριθμος Perceptron ενός στρώματος

Βήματα

1) Αρχικοποιούμε τα βάρη και το κατώφλι. Τα βάρη μπορούν να αρχίσουν από αρχική τιμή $w_i(0) = 0$ για κάθε i ή με μια μικρή τυχαία τιμή
2) Για κάθε στοιχειό j στο σετ εκπαίδευσης Z κάνουμε τα ακόλουθα βήματα
   2.1 Υπολογίζουμε την έξοδο:

$$y_j(t) = f\left[\boldsymbol{w}(t) \cdot \boldsymbol{x}_j\right] = f\left[w_0(t) + w_1(t)x_{j,1} + \ldots\ldots\ldots + w_n(t)x_{j,n}\right]$$

   2.2 Ενημερώνουμε τα βάρη:

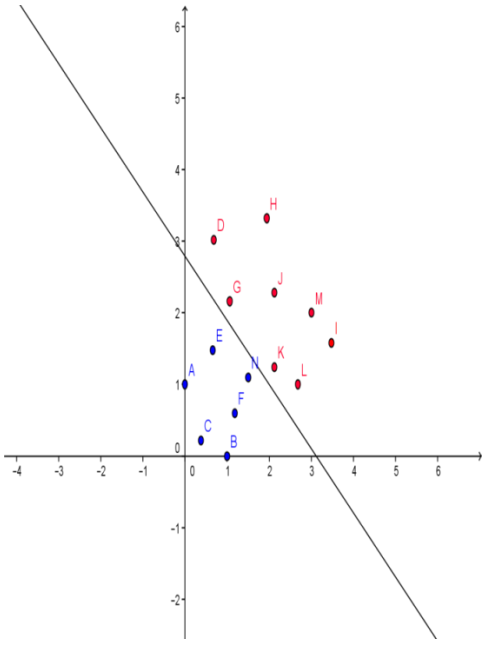
$$w_i(t+1) = w_i(t) + a\left(z_j - y_i(t)\right)x_{j,i}$$

3) Επαναλαμβάνουμε το 2 μέχρι το επαναληπτικό λάθος $\frac{1}{2}\sum_{j=1}^{n}(d_j - y_j(t))$ να είναι μικρότερο από το κατώφλι γ που θέσαμε ή μέχρι ο προκαθορισμένος αριθμός επαναλήψεων να έχει ολοκληρωθεί

Όπως αναφέραμε και πριν ο αλγόριθμος του perceptron που μόλις είδαμε συγκλίνει σε πεπερασμένο αριθμό επαναλήψεων μόνο αν τα "θετικά" παραδείγματα μπορούν να χωριστούν από τα "αρνητικά" παραδείγματα από ένα υπερεπίπεδο. Δηλαδή αν υπάρχει γ>0 και **w** τέτοιο ώστε (**w**.$\boldsymbol{x_j}$ + b) $d_j > \gamma$ για κάθε 1<j<m . Αν δεν υπάρχει διαχωριστικό υπερεπίπεδο τότε ο αλγόριθμος θα επαναλαμβάνεται μέχρι να φτάσει στον μέγιστο αριθμό βημάτων.

2.1.6.1.1.1. Παραδείγματα απλού Perceptron

Για αρχή θα δούμε ένα σχηματικό παράδειγμα για την εικόνα που θα έχει η λύση μας σε ένα πρόβλημα όπου τα δεδομένα μας είναι διαχωρίσιμα και στη συνέχεια θα κάνουμε μια απλή εφαρμογή στο πρόγραμμα python.



Γράφημα 1: Διαχωρισμός απλού Perceptron στο επίπεδο

Σημεία training set (x1,x2)	z
A	0
B	0
C	0
F	0
N	0
E	0
G	1
K	1
L	1
I	1
M	1
J	1
D	1
H	1

Πίνακας 1: Δεδομένα γραφικής εφαρμογής

Σε αυτό το παράδειγμα έχουμε διάφορα σημεία στο R^2 που αντιπροσωπεύουν τα διανύσματα του σετ εκμάθησης. Αν εφαρμόζαμε τον perceptron εδώ θα μας έβγαζε ως αποτέλεσμα μία συνάρτηση της μορφής της ευθείας που βλέπουμε η οποία προφανώς ως υπερεπίπεδο του R^2 διαχωρίζει όλες τις τιμές με εξαρτημένη μεταβλητή 0 με όλες που έχουν εξαρτημένη τιμή 1. Επίσης, πλέον μπορούμε να χωρίσουμε (όχι απαραίτητα ορθά) οποιαδήποτε νέα μεταβλητή πάρουμε και να βρούμε την προβλεπόμενη τιμή εξόδου της.

Παράδειγμα με εφαρμογή στο Python:

Για να καταλάβουμε πλήρως το πώς τρέχει ο αλγόριθμος του perceptron συμπεριλαμβάνουμε μια εφαρμογή στο πρόγραμμα python:

Έχουμε τα εξής δεδομένα για σετ εκπαίδευσης:

x_1	x_2	x_3	Z
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Πίνακας 2: Δεδομένα εφαρμογής στο Python

Θέτουμε

threshold = 0.5

learning_rate = 0.1

weights = [0, 0, 0]

training_set = [((1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1), ((1, 1, 1), 0)]

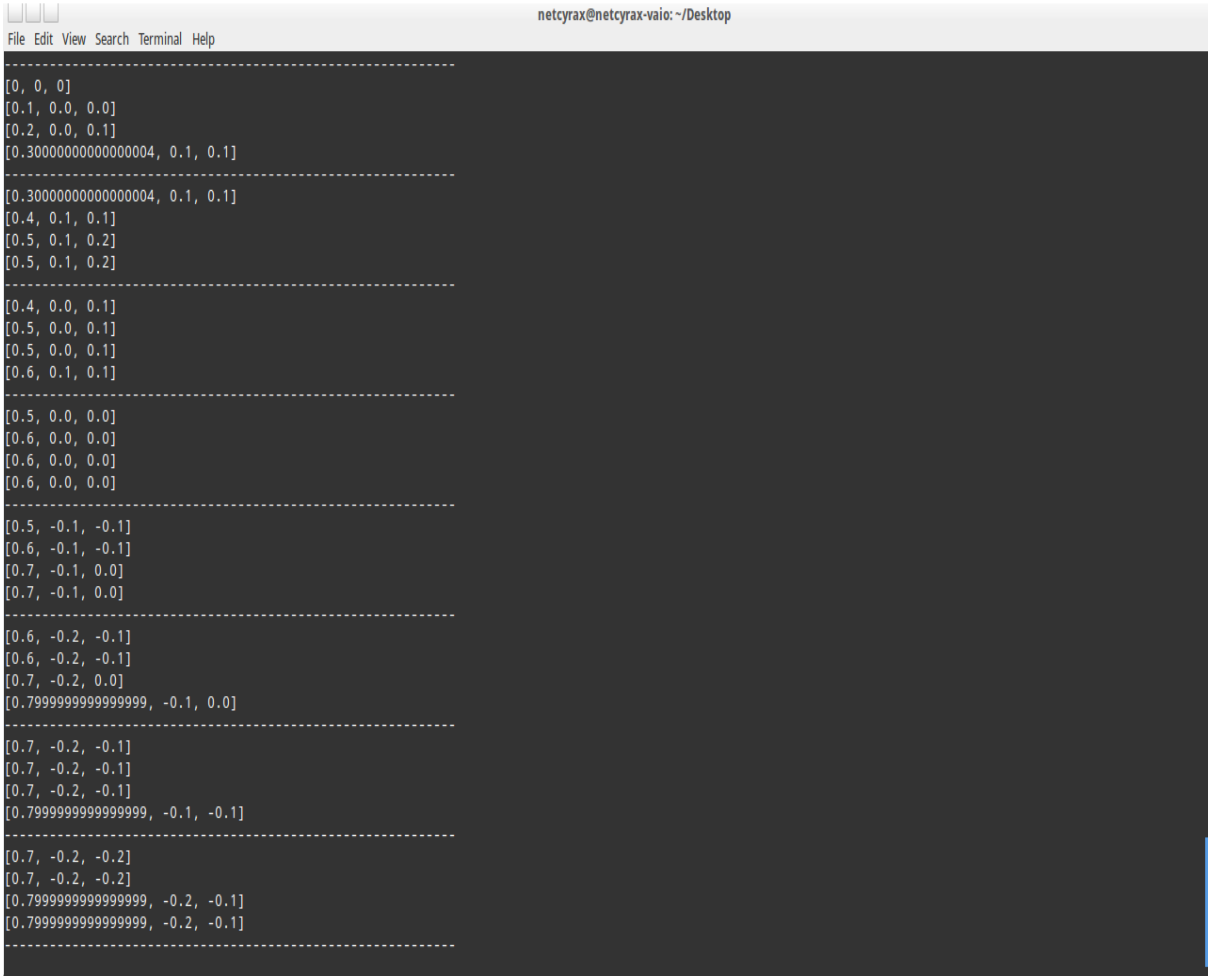
και ο αλγόριθμος να μας επιστρέφει

```
def dot_product(values):
   return sum(value * weight for value, weight in zip(values, weights))
```

Γράφουμε τον αλγόριθμό μας:

```
while True:
   print '-' * 60
   error_count = 0
   for input_vector, desired_output in training_set:
      print weights
      result = dot_product(input_vector) > threshold
      error = desired_output - result
      if error != 0:
         error_count += 1
         for index, value in enumerate(input_vector):
            weights[index] += learning_rate * error * value
   if error_count == 0:
      break
```

και λαμβάνουμε τα αποτελέσματα:



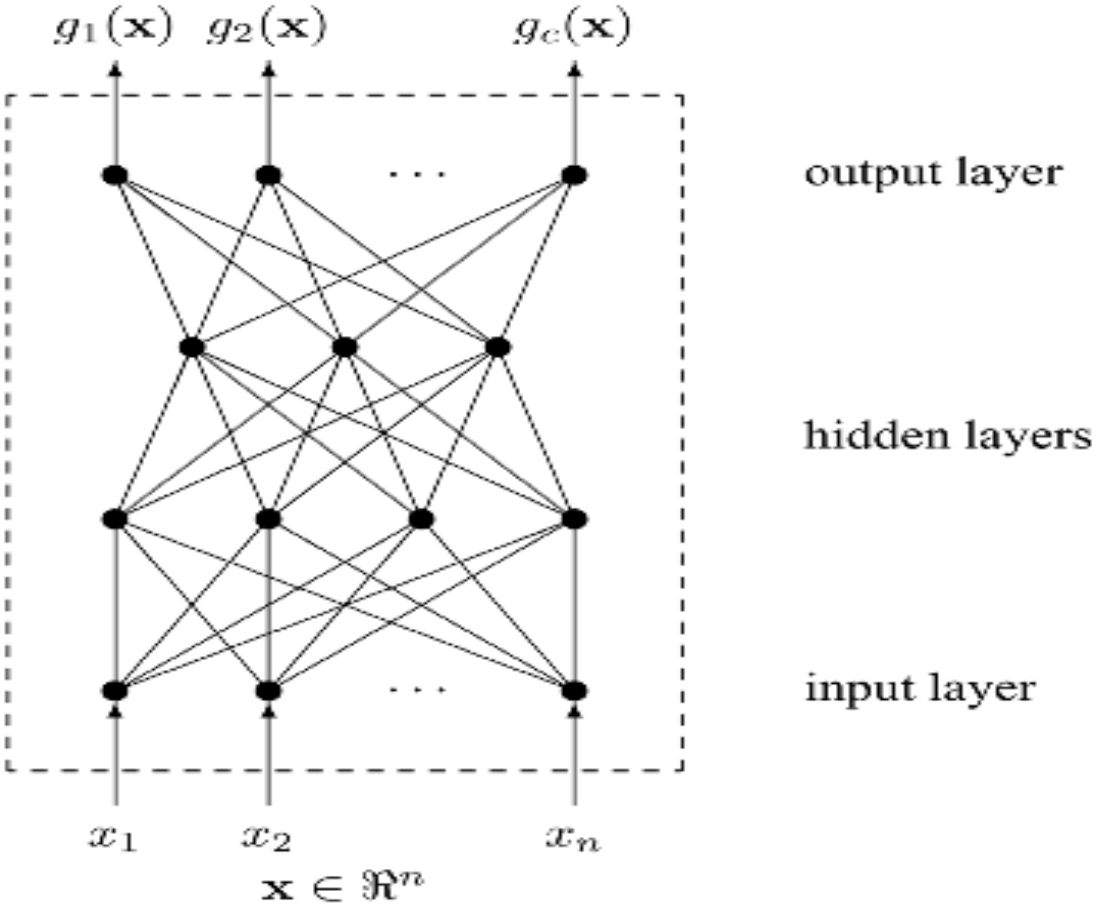
<u>Εικόνα 5</u>: Perceptron ενός στρώματος στο Python

Βλέπουμε ότι ο αλγόριθμος συγκλίνει και τα τελικά βάρη είναι **w=(**0,8, -0.2, -0.1)

Αν και τα απλά Perceptron λύνουν όλα τα διαχωρίσιμα προβλήματα τα περισσότερα προβλήματα δεν είναι διαχωρίσιμα και άρα δεν μπορούν να λυθούν με το απλό perceptron. Άρα τα απλά perceptron δεν εκμεταλλεύονται πλήρως τα νευρωνικά δίχτυα. Για αυτό το λόγο κρίνεται και αναγκαία η επέκταση του σε perceptron πολλών στρωμάτων (Multi Layer perceptron - MLP)

2.1.6.2 Perceptron πολλών στρωμάτων:

Ο perceptron πολλών στρωμάτων είναι η σύνδεση πολλών perceptron ενός στρώματος μαζί. Αυτό καλείται μια μορφή εμπροσθοτροφοδότησης (feed forward) γιατί η έξοδος του στρώματος εισόδου και όλα τα ενδιάμεσα στρώματα υποβάλλονται μόνο στο υψηλότερο στρώμα κάθε φορά.



Εικόνα 6: Perceptron πολλών στρωμάτων

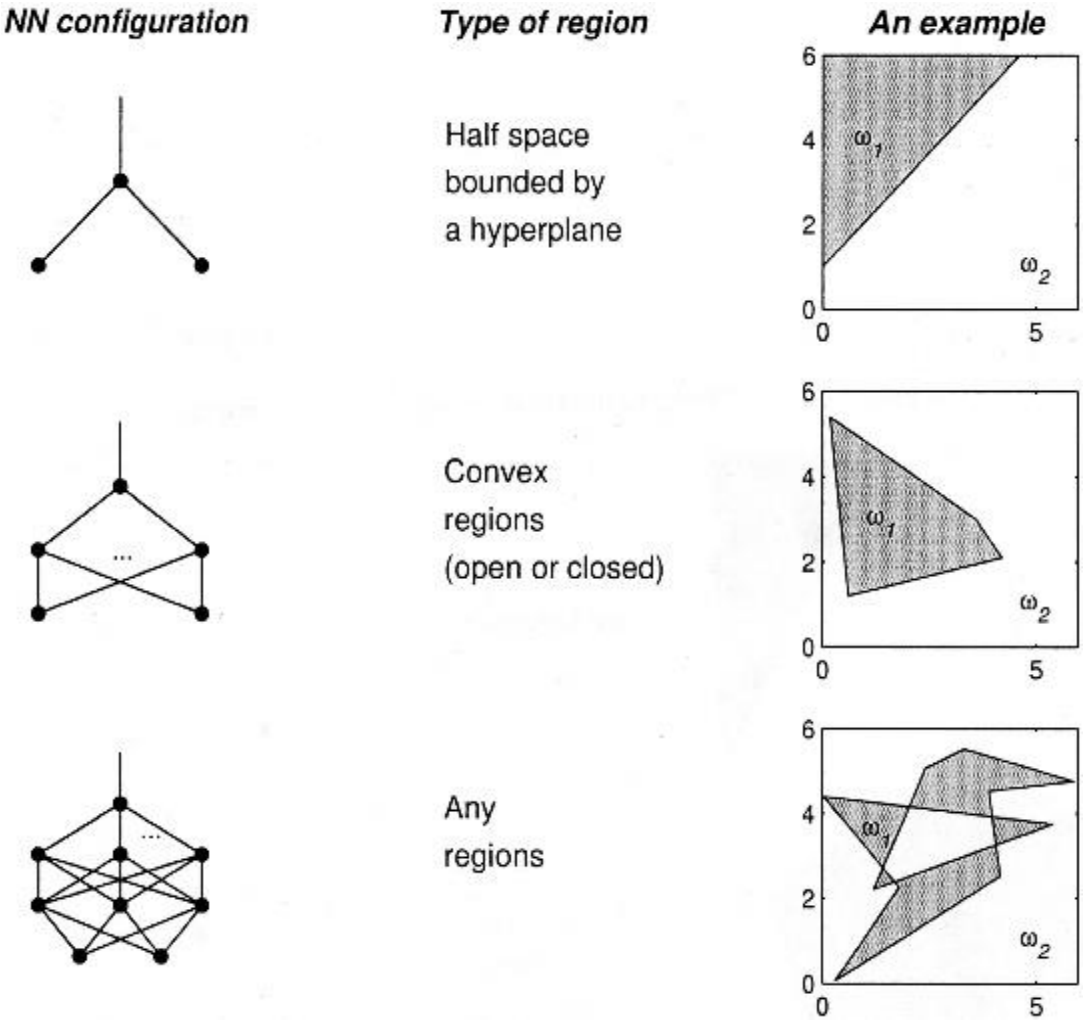
Εδώ η λέξη στρώμα εννοεί μια σειρά perceptron. Το διάνυσμα **x** υποβάλλεται στο στρώμα εισόδου (άρα και ο αριθμός των perceptron στο στρώμα εισόδου εξαρτάται από τον αριθμό των εισόδων που θέλουμε να δώσουμε). Στη συνέχεια μεταφέρεται σε ένα μη περιορισμένο αριθμό κρυμμένων στρωμάτων που έχουν ως στόχο να κωδικοποιούν τις εισόδους και να καθορίζουν τις εξόδους του διχτύου. Τέλος στο στρώμα εξόδου που αποτελείται από C διαφορετικές συναρτήσεις $\{g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots\ldots, g_c(\boldsymbol{x})\}$ παρουσιάζεται η έξοδος του όλου διχτύου.

Για τον MLP υπάρχουν κάποιες συμβάσεις που πρέπει να γίνονται έτσι ώστε να λειτουργήσει σωστά:

- Η συνάρτηση ενεργοποίησης στο στρώμα εισόδου είναι η ταυτοτική συνάρτηση
- Δεν υπάρχουν πλευρικές ενώσεις μεταξύ στρωμάτων του ίδιου στρώματος (για αυτό και ο όρος εμπροσθοτροφοδότηση)
- Μη γειτονικά στρώματα δεν συνδέονται άμεσα
- Όλοι οι νευρώνες σε όλα τα κρυμμένα στρώματα έχουν την ίδια συνάρτηση ενεργοποίησης

Σημείωση: Το μοντέλο MLD είναι το πιο διαδεδομένο νευρωνικό δίχτυο και τα περισσότερα αποτελέσματα θεωρίας αναπτύσσονται για αυτό το μοντέλο. Τα δύο επόμενα σημεία αναφέρονται λόγω της θεωρητικής τους σημασίας αν και πρακτικά από τη στιγμή που δεν μας λένε πώς να κατασκευάσουμε το MLP δεν μας βοηθάνε

- Στο τέλος της δεκαετίας του 1980 αποδείχτηκε ότι το MLP με 2 κρυμμένα στρώματα από νευρώνες μπορεί να προσεγγίσει οποιαδήποτε περιοχή ταξινόμησης στο R^n με συγκεκριμένη ακρίβεια.
- Αργότερα αποδείχτηκε ότι ακόμα και τα MLP με ένα κρυμμένο στρώμα προσεγγίζουν οποιαδήποτε συνάρτηση με οποιαδήποτε ακρίβεια.



<u>Σχήμα 8</u>: Πιθανές περιοχές ταξινόμησης MLP με ένα, δύο ή τρία κρυμμένα στρώματα

2.2 Λογιστική παλινδρόμηση

Η λογιστική παλινδρόμηση είναι μία ειδική περίπτωση των γενικευμένων γραμμικών μοντέλων. Η διαφορά του με την γραμμική παλινδρόμηση είναι ότι ενώ στη γραμμική παλινδρόμηση η εξαρτημένη μεταβλητή (y) είναι συνεχής, στη λογιστική παλινδρόμηση είναι κατηγορηματική. Πιο συγκεκριμένα στη συνέχεια θα υποθέσουμε ότι η εξαρτημένη μεταβλητή είναι δίτημη (y ϵ {0,1}) . Με λίγα λόγια η πρόβλεψη είναι αποτέλεσμα μιας διαδικασίας Bernoulli δηλαδή επιτυχία /αποτυχία, θάνατος ασθενή/ επιβίωση, άνοδος τιμής μετοχής/κάθοδος κτλ. Στην περίπτωση που δεν έχουμε δίτημη εξαρτημένη μεταβλητή ακολουθούμε την μέθοδο της πολυωνιμικής λογιστικής παλινδρόμησης (multinomial logistic regression) που ξεφεύγει από τους σκοπούς της παρούσας εργασίας και δεν θα μελετηθεί.

2.2.1 Μοντέλο Λογιστικής παλινδρόμησης

Ας υποθέσουμε ότι έχουμε μία πειραματική εκτέλεση με δίτιμη απόκριση y (y ϵ {0 , 1}), η οποία εξαρτάται από ένα σύνολο μεταβλητών παλινδρόμησης $x_1, x_2, \ldots, x_m$. Οι μεταβλητές αυτές μπορεί να είναι για παράδειγμα το ύψος ή η ηλικία ενός ασθενή και η απόκριση να είναι το αν ενεργεί ένα φάρμακο ή όχι σε αυτόν, αν ο άνεργος βρίσκει εργασία ή όχι. Εάν ορίσουμε την τιμή $y = 1$ σαν επιτυχία με πιθανότητα p και την τιμή $y = 0$ σαν αποτυχία με πιθανότητα $1 - p$, τότε μπορούμε να πούμε ότι η y είναι τ.μ της κατανομής Bernoulli, με $E(y) = p$ και $V(y) = p(1 - p)$.

Επεκτείνουμε τώρα σε μια σειρά από n-δοκιμές. Ορίζουμε την τυχαία μεταβλητή y να είναι ο αριθμός επιτυχιών σε n-δοκιμές ($y = 0,1,2, \ldots, n$), υπό την υπόθεση ότι η πιθανότητα επιτυχίας p είναι ίδια σε κάθε δοκιμή και με τις δοκιμές να είναι ανεξάρτητες μεταξύ τους. Τότε ισχύει η Διωνυμική (binomial) κατανομή: $y \sim b(n, p)$ με συνάρτηση πιθανότητας: $f(y) = \binom{n}{y} p^y (1 - p)^{n-y}$, όπου η πιθανότητα επιτυχίας p είναι η παράμετρος της Διωνυμικής αυτής κατανομής, που είναι η κατάλληλη κατανομή προς περιγραφή της y σε τέτοια περίπτωση, με $E(y) = np$ και $V(y) = np(1 - p)$.

Σε πολλές περιπτώσεις που έχουμε δίτιμη μεταβλητή απόκρισης y (με τιμές 0 ή 1), μπορεί να εξαρτάται από επεξηγηματικές μεταβλητές $\boldsymbol{x}$ (ανεξάρτητες ή συμμεταβλητές). Η εξάρτηση αυτή εισάγεται μέσω της εξάρτησης της πιθανότητας επιτυχίας p από τις $\boldsymbol{x}$ (π.χ. η πιθανότητα να ενεργήσει σε κάποιο ασθενή ένα φάρμακο να εξαρτάται από τη θερμοκρασία, το φύλο, την ηλικία κλπ). Πιο συγκεκριμένα κατασκευάζεται το μοντέλο της Λογιστικής Παλινδρόμησης, μέσω της συνάρτησης σύνδεσης του [12]:

$$\eta_x = g\left(E(y_x)\right) = g(\mu_x) = \boldsymbol{x}'\boldsymbol{\beta} \qquad (2.2.1.1)$$

με την ακόλουθη δομή:

- $y_x \sim b(n_x, p_x)(n_x > 1,\ διωνυμικά\ δεδομένα)$
  ή $y_x \sim B(\mu_x)(n_x = 1, δυαδικά\ δεδομένα)$
- $\eta_x = g\ (\mu_x) = \ln\frac{\mu_x}{n_x - \mu_x} = \ln\frac{p_x}{1-p_x} = logit(p_x) = \boldsymbol{x}'\boldsymbol{\beta}\ (συνάρτηση\ logit)$
- ανεξαρτησία μεταξύ των παρατηρήσεων y_x,

όπου n_x ο αριθμός των επαναλήψεων της τιμής του διανύσματος $\boldsymbol{x}$ των επεξηγηματικών μεταβλητών.

Τώρα, αν αντιστρέψουμε τη συνάρτηση σύνδεσης προκύπτει:

$$p_{\boldsymbol{x}} = e^{\eta_x}/(1 + e^{\eta_x}), \qquad με\ περιορισμό\ \ 0 < p_{\boldsymbol{x}} < 1 \qquad (2.2.1.2)$$

Για κάθε i-παρατήρηση το μοντέλο γράφεται:

$$ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}, \qquad i = 1, \ldots, n$$

με πιθανότητα επιτυχίας:

$$p_i = p_{\boldsymbol{x}_i} = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik})} \qquad (2.2.1.3)$$

και άρα:

$$\mathrm{E}(y_i) = n_i p_i = n_i \frac{e^{\boldsymbol{x}_i'\boldsymbol{\beta}}}{1 + e^{\boldsymbol{x}_i'\boldsymbol{\beta}}} \ .$$

Η πιο πάνω συνάρτηση σύνδεσης που χρησιμοποιείται είναι η ***logit*** και αποτελεί την πιο συνηθισμένη επιλογή για δεδομένα που ακολουθούν τη Διωνυμική κατανομή. Άλλες συναρτήσεις σύνδεσης που χρησιμοποιούνται είναι:

- $g(\mu_x) = \ln[-\ln(1 - p_x)] = \boldsymbol{x}'\boldsymbol{\beta}$ $(συνάρτηση\ complementary\ log - log)$
- $g(\mu_x) = \Phi^{-1}(p_x) = \boldsymbol{x}'\boldsymbol{\beta}$ $(συνάρτηση\ probit)$.

2.2.2 Παράδειγμα Μοντέλου Λογιστικής Παλινδρόμησης

Με χρήση ενός απλουστευμένου μοντέλου θα εκτιμήσουμε την πιθανότητα να πεθάνει κάποιος ασθενής, σε διάστημα 10 χρόνων, από ένα καρδιακό νόσημα. Το μοντέλο περιλαμβάνει μόνο τρείς παράγοντες (επεξηγηματικές μεταβλητές), την ηλικία, το φύλο και το επίπεδο της χοληστερόλης στο αίμα του ασθενή. Οι παράμετροι του μοντέλου είναι οι εξής:

β_0 = -5.0, β_1 = +2.0, β_2 = -1.0, β_3 = +1.2

x_1 = η ηλικία σε χρόνια, πάνω από τα 50

x_2 = το φύλο (0: αρσενικό και 1: θηλυκό)

x_3 = το επίπεδο χοληστερόλης σε mmol/L, πάνω απο το 5.0

p = η πιθανότητα θανάτου του ασθενή (0: πέθανε και 1: ακόμα ζωντανός)

Μπορούμε να εκφράσουμε το μοντέλο ως εξής:

$$p = 1/(1 + e^{-z}) \text{ όπου } z = -5.0 + 2.0x_1 - 1.0x_2 + 1.2x_3$$

Σε αυτό το μοντέλο, η πιθανότητα θανάτου αυξάνεται με την ηλικία (το *z* αυξάνεται κατά 2.0 για κάθε επιπλέον έτος πάνω από τα 50), μειώνεται εάν έχουμε θηλυκό ασθενή (το *z* μειώνεται κατά 1.0 στην περίπτωση θηλυκού ασθενή) και αυξάνεται με αυξημένο επίπεδο της χοληστερόλης (το *z* αυξάνεται κατά 1.2 για κάθε 1 mmol/L παραπάνω από τα 5 mmol/L). Θα χρησιμοποιήσουμε το παραπάνω μοντέλο σε έναν άντρα ασθενή 50 χρονών με επίπεδο χοληστερόλης 7.0 mmol/L.

Η πιθανότητα να πεθάνει είναι:

$$p = 1/(1 + e^{-z})$$

$$όπου\ z = -5.0 + 2.0(50 - 50) - (1.0)0 + 1.2(7.0 - 5.0) => z = -2.6$$

Από το αποτέλεσμα υπολογίζουμε ότι η πιθανότητα να πεθάνει ο συγκεκριμένος ασθενής από καρδιακό νόσημα, σε διάστημα 10 χρόνων, είναι 0.069 ή αλλιώς 7%.

2.2.3 Εκτίμηση των Συντελεστών Παλινδρόμησης

Η λογιστική παλινδρόμηση χρησιμοποιείται για την εξαγωγή συμπερασμάτων σε πολλές διαφορετικές περιπτώσεις, όπως για παράδειγμα σε κλινικές δοκιμές όπου πρέπει να συγκρίνουμε τα αποτελέσματα διαφορετικών θεραπειών των οποίων το αποτέλεσμα έχει δυαδική μορφή. Για τη βελτίωση του μοντέλου δοκιμάζεται η σημασία της κάθε μεταβλητής.

Λόγω της λογιστικής παλινδρόμησης, έχουμε τη δυνατότητα ερμηνείας των τιμών των συντελεστών $\hat{\beta}$, αλλά και των διαστημάτων εμπιστοσύνης τους. Εφόσον εκτιμηθούν οι συντελεστές αυτοί, η σχέση μεταξύ της προσαρμοσμένης πιθανότητας απόκρισης $\hat{p}$ και των τιμών των $x_0,\ x_1,\ x_2$ (επεξηγηματικών μεταβλητών), εκφράζεται ως:

$$\hat{p} = \frac{e^{x'\hat{\beta}}}{1 + e^{x'\hat{\beta}}}$$

ή ισοδύναμα μέσω του λόγου των συμπληρωματικών ή σχετικών πιθανοτήτων (odds), χρησιμοποιώντας τη συνάρτηση σύνδεσης $logit(p)$:

$$\ln\left(\frac{\hat{p}}{1-\hat{p}}\right) = \boldsymbol{x}'\widehat{\boldsymbol{\beta}} => \frac{\hat{p}}{1-\hat{p}} = e^{x'\widehat{\beta}} = \exp\left(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_k x_k\right), \qquad \text{όπου } x_0 \equiv 1$$

Από το *odds* προκύπτει ότι αν ο εκτιμημένος συντελεστής $\widehat{\beta_j}$ είναι θετικός, ο παράγοντας $e^{\widehat{\beta_j}} > 1$, γεγονός που σημαίνει πως το odds αυξάνεται με την αύξηση της x_j. Αντίθετα αν ο $\widehat{\beta_j}$ είναι αρνητικός, ο παράγοντας $e^{\widehat{\beta_j}} < 1$ και η σχετική πιθανότητα μειώνεται με την αύξηση της x_j .

Οι παράμετροι της παλινδρόμησης μπορούν να εκφραστούν και μέσα από το λόγο του λόγου των συμπληρωματικών πιθανοτήτων, δηλαδή μέσα από το λόγο των odds (odds ratio).

Παράδειγμα:

Ας υποθέσουμε για παράδειγμα, ένα μοντέλο με δύο συμμεταβλητές x_1 και x_2. Η μεταβλητή x_1 είναι κατηγορική και ότι μια ομάδα από τα πειραματικά μας υποκείμενα μπορούν να χωριστούν σε αυτά που τους χορηγήθηκε μια δόση από βιταμίνη C ($x_1 = 0$) και σε αυτά που δεν τους χορηγήθηκε τίποτα ($x_1 = 1$). Επίσης, έχουμε τη x_2 ως ποσοτική μεταβλητή (π.χ. θερμοκρασία). Οπότε και η απόκριση y, που ισούται με την πιθανότητα να έχει μολυνθεί το αναπνευστικό σύστημα ή όχι, παίρνει τις τιμές $y = 1$ και $y = 0$, αντίστοιχα. Αν χρησιμοποιήσουμε την πιο πάνω σχέση θα έχουμε, για το παράδειγμα:

$$\ln\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$$

Τώρα, αν θεωρήσουμε ένα υποκείμενο στο οποίο χορηγείται η βιταμίνη C ($x_1 = 0$), τότε η $\exp(\beta_0)$ μπορεί να μεταφραστεί σαν το λόγο συχνοτήτων για τα υποκείμενα που μολύνθηκαν προς αυτά που δε μολύνθηκαν, για όλο τον πληθυσμό. Όσον αφορά αυτά που δε χορηγήθηκε βιταμίνη C ($x_1 = 1$), θα έχουμε όπως πιο πάνω αλλά με $x_1 = 1$.

Χρησιμοποιώντας την πιο πάνω ερμηνεία $\hat{\beta}_0$ βρίσκουμε την ερμηνεία για το $\hat{\beta}_1$. Για την ομάδα που δε δέχτηκε τη θεραπεία ισχύει:

$$\frac{\frac{\hat{p}_1}{1-\hat{p}_1}}{\frac{\hat{p}_2}{1-\hat{p}_2}} = \frac{odds(y=1|x_1=1,x_2)}{odds(y=0|x_1=0,x_2)} = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 + \hat{\beta}_2 x_2)}{\exp(\hat{\beta}_0 + \hat{\beta}_2 x_2)} = e^{\widehat{\beta}_1}$$

Άρα, η ποσότητα $\exp(\hat{\beta}_1)$ μπορεί να ερμηνευτεί σαν το λόγο των συχνοτήτων της ομάδας που δε δέχτηκε θεραπεία, σε σχέση με αυτή που δέχτηκε και είναι ανεξάρτητη της x_2. Προφανώς, ένας ερευνητής ερμηνεύει μια τιμή $\hat{\beta}_0 \ll 0$, όπως επίσης και μια τιμή $\hat{\beta}_1 \gg 0$ να είναι ευνοϊκή (στατιστικά σημαντική) για τη θεραπεία.

2.2.4 Μέθοδος της Μέγιστης Πιθανοφάνειας

Όπως σε όλα τα γενικευμένα γραμμικά μοντέλα, η προσαρμογή του μοντέλου στα δεδομένα γίνεται με τη μέθοδο της μέγιστης πιθανοφάνειας. Ας υποθέσουμε ότι τα δεδομένα μας είναι χωρισμένα σε κατηγορίες. Δηλαδή, έχουμε n_i το πλήθος πειραματικές μονάδες στο i-οστό σημείο δεδομένων (για παράδειγμα, μπορούμε να θεωρήσουμε ότι το n_i είναι το πλήθος των πειραματόζωων στα οποία έχουμε παρέχει μια συγκεκριμένη δοσολογία φαρμάκου). Άρα έχουμε το δείγμα με τιμές $y_1, y_2, \dots, y_m$, με μέσες τιμές $E(y_i) = \mu_i = n_i p_i$, $V(y_i) = n_i p_i(1-p_i)$ και συμμεταβλητές $\boldsymbol{x}_i{}' = (x_{i0}, x_{i1}, \dots, x_{ik})$, όπου n_i ο αριθμός δοκιμών της στατιστικής μονάδας-i και $\sum_{i=1}^{m} n_i = n$, p_i η αντίστοιχη πιθανότητα επιτυχίας και $x_{i0} = 1$.

Συνεπώς, υποθέτοντας κανονική συσχέτιση $n_i = g(\mu_i) = \boldsymbol{x}_i{}'\boldsymbol{\beta}$ στα GLMs, η συνάρτηση πιθανοφάνειας του δείγματος γράφεται [42]:

$$L(\beta) = \prod_{i=1}^{n} \binom{n_i}{y_i} p_i^{y_i}(1-p_i)^{n_i - y_i} \qquad (2.2.4.1)$$

Η πιθανοφάνεια εξαρτάται από τις άγνωστες πιθανότητες επιτυχίας p_i, οι οποίες με τη σειρά τους εξαρτώνται από τα $\boldsymbol{\beta}$ μέσω της εξίσωσης 2.2.

Έτσι η συνάρτηση πιθανοφάνειας μπορεί να θεωρηθεί ως συνάρτηση των $\boldsymbol{\beta}$ με:

$$l = \ln L(\boldsymbol{\beta}) \ = \sum_{i=1}^{n} \left\{ ln \binom{n_i}{y_i} + y_i x_i{}'\boldsymbol{\beta} - n_i \ln(1 + e^{x'_i \boldsymbol{\beta}}) \right\} \qquad (2.2.5.1)$$

Παραγωγίζοντας έχουμε:

$$\frac{\partial \ln L(\beta)}{\partial \beta_j} \ = \sum_{i=1}^{n} (y_i - n_i p_i) x_{ij} \qquad (2.2.6.1)$$

Οι εκτιμήτριες μέγιστης πιθανοφάνειας των β_j προκύπτουν με την ικανοποίηση των εξισώσεων:

$$\sum_{i=1}^{n}(y_i - n_i\hat{p}_i)x_{ij} = \sum_{i=1}^{n}(y_i - \hat{\mu_i})x_{ij} = 0, \quad j = 0,1,\dots,k$$

$$=> \boldsymbol{X}' = (\boldsymbol{y} - \widehat{\boldsymbol{\mu}}) = \mathbf{0} \qquad (2.2.6.2)$$

όπου $\widehat{\boldsymbol{\mu}}' = (\hat{\mu}_1, \hat{\mu}_2, \dots, \hat{\mu}_n)$ με $ln\hat{\mu}_i = \boldsymbol{x_i}'\widehat{\boldsymbol{\beta}}$. Η εξίσωση είναι μη γραμμική ως προς τα $\widehat{\boldsymbol{\beta}}$, επειδή $ln\hat{\mu}_i = \exp(\boldsymbol{x_i}'\widehat{\boldsymbol{\beta}})$, επομένως για να λύσουμε το σύστημα θέλουμε μια επαναληπτική διαδικασία. Μια τέτοια διαδικασία είναι αυτή των σταθμισμένων ελαχίστων τετραγώνων. Έτσι μπορούμε να υπολογίσουμε τις εκτιμήτριες $\widehat{\boldsymbol{\beta}}$ για να καταλήξουμε στα συμπεράσματα που επιθυμούμε, διότι οι ποσότητες $\exp(\hat{\beta}_j)$ εκφράζουν την αναμενόμενη πολλαπλασιαστική μεταβολή της y για μία μονάδα αύξησης της αντίστοιχης x_j, κρατώντας τις υπόλοιπες συμμεταβλητές σταθερές.

2.2.4.1 Μέθοδος Σταθμισμένων Ελαχίστων τετραγώνων

Για να λύσουμε την εξίσωση (2.2.6.2) και να βρούμε τα $\widehat{\boldsymbol{\beta}}$ με τη μέθοδο των σταθμισμένων ελαχίστων τετραγώνων έχουμε τον τύπο των υπολοίπων

$$S = \sum_{\mathrm{i=1}}^{\mathrm{m}}\left[\frac{(\mathrm{y_i}-\mu_\mathrm{i})^2}{\sigma_\mathrm{i}^{\,2}}\right] \qquad (2.2.4.1.1)$$

Όπου $\mu_i = n_i P(\boldsymbol{x_i})$ και ${\sigma_i}^2$είναι η διωνυμική διακύμανση στο i-οστό σημείο δεδομένων με

$${\sigma_i}^2 = n_i P(\boldsymbol{x_i})[1 - P(\boldsymbol{x_i})] = n_i \frac{e^{-\boldsymbol{x_i}'\boldsymbol{\beta}}}{(1 + e^{-\boldsymbol{x_i}'\boldsymbol{\beta}})^2}$$

Στη συνέχεια ελαχιστοποιούμε το S και κατ' επέκταση την 4.2.4.1.1 όπου η διακύμανση $({\sigma_i}^2)$ είναι σταθερή, επομένως παραγωγίζουμε μόνο τον αριθμητή του S και παίρνουμε:

$$2\left[\frac{\sum_{i=1}^{m}(y_i - \mu_i)}{{\sigma_i}^2}\right](\frac{\partial \mu_i}{\partial \boldsymbol{\beta}})$$

με:

$$\frac{\partial \mu_i}{\partial \boldsymbol{\beta}} = n_i P(\boldsymbol{x_i})[1 - P(\boldsymbol{x_i})]\boldsymbol{x_i} = {\sigma_i}^2 \boldsymbol{x_i}$$

Επομένως , η λύση που παίρνουμε από την ελαχιστοποίηση του σταθμισμένου αθροίσματος τετραγώνων των υπολοίπων με σταθερό ${\sigma_i}^2$είναι:

$$\sum_{i=1}^{m} (y_i - \mu_i)\boldsymbol{x_i} = 0$$

2.2.5 Έλεγχος Σημαντικότητας Συντελεστών

Μετά των υπολογισμό των παραμέτρων, αυτό που μας ενδιαφέρει για το μοντέλο μας είναι η σημαντικότητα των παραμέτρων που υπολογίσαμε. Αυτό επιτυγχάνεται ελέγχοντας μια στατιστική υπόθεση για να διευκρινίσουμε αν οι ανεξάρτητες μεταβλητές μας είναι σημαντικά συσχετισμένες με την μεταβλητή απόκρισης. Η μέθοδος για να κάνουμε αυτούς τους ελέγχους είναι σχετικά γενικές και διαφέρουν μεταξύ των μοντέλων μόνο σε συγκεκριμένες λεπτομέρειες.

Η προσέγγιση μας για τον έλεγχο σημαντικότητας του συντελεστή των μεταβλητών είναι κάνοντας την ερώτηση : *"Το μοντέλο που περιέχει τη μεταβλητή μας λέει περισσότερα για το αποτέλεσμα παρά αυτό που δεν την περιέχει?"* Σαφώς η απάντηση βρίσκεται στη σύγκριση των παρατηρουμένων μεταβλητών απόκρισης των 2 μοντέλων (αυτού που περιέχει την μεταβλητή και αυτού που δεν την περιέχει).

2.2.5.1. Έλεγχος με τη χρήση μέγιστης πιθανοφάνειας

Στα γραμμικά μοντέλα συγκρίνουμε τη διαφορά στο άθροισμα τετραγώνων σφάλματος (έστω D)

G=D(μοντέλο χωρίς τις υπό εξέταση μεταβλητές) – D(μοντέλο με όλες τις μεταβλητές)

Αντίστοιχα στη λογιστική παλινδρόμηση η διαφορά στο άθροισμα τετραγώνων του σφάλματος αντικαθίσταται, από τη διαφορά της log πιθανοφάνειας:

$$-2\ln\frac{L(\mu o \nu \tau \acute{\varepsilon} \lambda o\ \chi \omega \rho \acute{\iota} \varsigma\ \tau \iota \varsigma\ \upsilon \pi \acute{o}\ \varepsilon \xi \acute{\varepsilon} \tau \alpha \sigma \eta\ \mu \varepsilon \tau \alpha \beta \lambda \eta \tau \acute{\varepsilon} \varsigma)}{L(\mu o \nu \tau \acute{\varepsilon} \lambda o\ \mu \varepsilon\ \acute{o} \lambda \varepsilon \varsigma\ \tau \iota \varsigma\ \mu \varepsilon \tau \alpha \beta \lambda \eta \tau \acute{\varepsilon} \varsigma)} = -2\{l(\ \boldsymbol{\beta}*) - l(\boldsymbol{\beta})\} \sim X_d^2$$

Με L η πιθανοφάνεια και d η διαφορά των παραμέτρων στα δύο μοντέλα.

Παράδειγμα:

Έχουμε το μοντέλο $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ και θέλουμε να εξετάσουμε την υπόθεση $H_0: \beta_1, \beta_3 = 0$. Οπότε θα έχουμε τον πιο κάτω έλεγχο:

$$2\{\ln(L(\beta_0, \beta_1, \beta_2, \beta_3) - \ln(L(\beta_0^*, \beta_2^*)\} \sim X_2^2$$

όπου $L(\beta_0^*, \beta_2^*)$ το μοντέλο για το οποίο έχουμε επικαλεστεί την πιο πάνω μηδενική υπόθεση. Έτσι αν η λογαριθμημένη πιθανοφάνεια αυξηθεί σημαντικά, έχουμε σοβαρές ενδείξεις για να απορρίψουμε τη μηδενική υπόθεση και να συμπεριλάβουμε όλες τις μεταβλητές στο μοντέλο, αφού τις θεωρούμε στατιστικά σημαντικές.

Άρα από τα πιο πάνω καταλήγουμε στο συμπέρασμα ότι η μέθοδος της μέγιστης πιθανοφάνειας είναι πολύ χρήσιμη για ελέγχους υποθέσεων που θα μας βοηθήσουν στην επιλογή του κατάλληλου μοντέλου στη λογιστική παλινδρόμηση.

2.2.5.2 Έλεγχος με τη μέθοδο του WALD

Η πρώτη εφαρμογή της μεθόδου Wald έχει να κάνει με έλεγχο υποθέσεων για κάθε ξεχωριστό συντελεστή του μοντέλου της λογιστικής παλινδρόμησης. Πιο συγκεκριμένα, θέλουμε να ελέγξουμε: H_0: $\beta_j = 0$, H_1: $\beta_j \neq 0$, *με* $j = 1, \dots, k$ και με το $\hat{\beta}_j$ να εμφανίζεται στη γραμμική πρόβλεψη $\boldsymbol{x}_i{}'\widehat{\boldsymbol{\beta}}$ του λογιστικού μοντέλου.

Με τη μέθοδο της μέγιστης πιθανοφάνειας (§ 2.2.5.1) υπολογίσαμε τους συντελεστές $\hat{\beta}_j$, με εκτιμημένες διασπορές $\hat{V}(\hat{\beta}_j)$ που κάθε μία είναι το j-οστό διαγώνιο στοιχείο του πίνακα $\boldsymbol{I}^{-1}(\widehat{\boldsymbol{\beta}})$, με αντίστοιχο $se(\hat{\beta}_j) = \{\hat{V}(\hat{\beta}_j)\}^{1/2} = \{\boldsymbol{I}^{-1}(\widehat{\boldsymbol{\beta}})_{jj}\}^{1/2}$.

Επομένως για έναν εκτιμητή μέγιστης πιθανοφάνειας ισχύει ότι:

$$z_j = \frac{\hat{\beta}_j - \beta_j}{se(\hat{\beta}_j)} \qquad (2.2.5.2.1)$$

οποίος ακολουθεί ασυμπτωτικά την τυπική κανονική κατανομή N(0,1) και έτσι ισχύει για τη μηδενική υπόθεση:

$$z_j{}^2 = \left(\frac{\hat{\beta}_j}{se(\hat{\beta}_j)}\right)^2 \qquad (2.2.5.2.2)$$

που ακολουθεί ασυμπτωτικά την X^2-κατανομή. Οι τιμές των ελεγχοσυναρτήσεων υπολογίζονται για κάθε συντελεστή των μεταβλητών και αφορούν την προσαρμογή του μοντέλου. Έτσι ελέγχουμε αν απορρίπτεται η H_0 έναντι της H_1. Η μηδενική υπόθεση απορρίπτεται αν η p-τιμή του πιο πάνω ελέγχου είναι πολύ μικρή που σημαίνει ότι η αντίστοιχη μεταβλητή x_j θα είναι στατιστικά σημαντική για την πρόβλεψη και θα πρέπει να περιληφθεί στο μοντέλο. Αν δεν απορρίπτεται η μηδενική υπόθεση, με βάση τον έλεγχο, σημαίνει ότι έχουμε σημαντικές ενδείξεις ότι η συγκεκριμένη μεταβλητή δεν είναι στατιστικά σημαντική για το μοντέλο. Επίσης σημαντική προϋπόθεση για να μην πέσουμε έξω στους ελέγχους αυτούς, είναι να ελέγξουμε πρώτα και τη συσχέτιση μεταξύ των μεταβλητών, αλλά και τη συσχέτιση της κάθε μεταβλητής με την απόκριση.

Μια δεύτερη μορφή της Wald συμπερασματολογίας έχει να κάνει με τον υπολογισμό του διαστήματος εμπιστοσύνης της διωνυμικής πιθανότητας για κάποια δοσμένα ή αυθαίρετα δεδομένα. Λόγω της ύπαρξης της γραμμικής πρόβλεψης $\boldsymbol{x}'\widehat{\boldsymbol{\beta}}$ στο λογιστικό μοντέλο, ακολουθείται μια εναλλακτική διαδικασία υπολογισμού των διαστημάτων εμπιστοσύνης. Μπορούμε να ορίσουμε ένα $100(1-a)\%$ διάστημα εμπιστοσύνης (δ.ε.) στο $p = 1/\left(1+e^{-\boldsymbol{x}'\boldsymbol{\beta}}\right)$, όπου $p = p(x_i) = y_i$ χρησιμοποιώντας ένα διάστημα εμπιστοσύνης στο $\boldsymbol{x}'\boldsymbol{\beta}$. Η γραμμική πρόβλεψη περιλαμβάνει προφανώς όρους που είναι γραμμικοί στο $\boldsymbol{\beta}$ και μπορούμε να εκμεταλλευτούμε το γεγονός ότι οι εκτιμήτριες $\widehat{\boldsymbol{\beta}}$, είναι ασυμπτωτικά κανονικές. Άρα, ένα άνω διάστημα εμπιστοσύνης για το $\boldsymbol{x}'\boldsymbol{\beta}$, παράγει ένα άνω διάστημα εμπιστοσύνης για το p. Έτσι μετά από πράξεις, καταλήγουμε στο ότι μπορούμε να κατασκευάσουμε ένα $100\,(1-a)\%$ δ.ε. για την παράμετρο β_j, ως $\hat{\beta}_j \pm z_{\alpha/2} se\left(\widehat{\beta}_j\right)$, με $j = 1, 2, \dots, k$. Από το δ.ε. αυτό μπορούμε να προσδιορίσουμε και ένα $100(1-a)\%$ δ.ε. $exp\left[\widehat{\beta}_j \pm z_{a/2} se\left(\widehat{\beta}_j\right)\right]$, $j = 1,2, \dots, k$ για το λόγο των odds (odds ratio). Με τα διαστήματα εμπιστοσύνης αυτά παίρνουμε ακόμα περισσότερες πληροφορίες για τις εκτιμήτριες, για να καταλήξουμε σε ακόμα καλύτερα συμπεράσματα.

2.2.6 Ελεγχοσυνάρτηση DEVIANCE

Μια σημαντική τεχνική που χρησιμοποιείται για τη σύγκριση και ανάπτυξη των στατιστικών μοντέλων είναι η ελεγχοσυνάτηση Deviance ή απόκλιση. Κατ' αρχάς, ένα μοντέλο λέγεται κορεσμένο ή πλήρες όταν έχει τόσες παραμέτρους όσες είναι και οι παρατηρήσεις του. Έχουμε δύο μοντέλα Μ και Μ* με εκτιμήτριες $\widehat{\boldsymbol{\beta}}$ και $\widehat{\boldsymbol{\beta}}$* αντίστοιχα, όπου Μ*⊂Μ με όλες τις μεταβλητές του Μ* να περιέχονται στο Μ, δηλαδή από το Μ απορρίψαμε κάποιες μη στατιστικά σημαντικές μεταβλητές.

Όπως έχουμε αναφέρει και προηγουμένως, για να συγκρίνουμε τα μοντέλα αυτά χρησιμοποιούμε τον έλεγχο με βάση το λόγο των μεγιστοποιημένων πιθανοφανειών

$$-2\{l(\ \widehat{\boldsymbol{\beta}}*\ ) - l(\widehat{\boldsymbol{\beta}})\} \sim X_d^2$$

με *H_0: ισχύει το μοντέλο Μ** έναντι της εναλλακτικής *H_1: ισχύει το μοντέλο Μ*.

Αν γράψουμε τώρα την εναλλακτική υπόθεση H_1 ως *H_S: το κορεσμένο μοντέλο με $p_1 = n$ (αριθμό παραμέτρων ίσο με τον αριθμό των παρατηρήσεων)* και την *H_0: το υποψήφιο μοντέλο με $p_0 = p < n$*, τότε οι προβλεπόμενες τιμές $\tilde{\mu}_i$ ισούνται με τις παρατηρούμενες y_i. Έτσι έχουμε τη deviance με $D_1 = -2(\hat{l}_1 - \hat{l}_S)$ και $D_2 = -2(\hat{l}_0 - \hat{l}_S)$ ως:

$$D_0 - D_1 = (\hat{l}_0 - \hat{l}_1) \sim X_d^2 \quad \text{ασυμπτωτικά} \qquad (2.2.6.1)$$

όπου $l(.)$: η λογαριθμοποιημένη συνάρτηση πιθανοφάνειας (του ελαττωμένου και του πλήρες μοντέλου) και d: η διαφορά του πλήθους των παραμέτρων του ελαττωμένου από του πλήρες μοντέλου.

Τώρα για το μοντέλο της λογιστικής παλινδρόμησης, μέσω της $ln\left(\frac{p_i}{1-p_i}\right) = \boldsymbol{x_i}'\boldsymbol{\beta}$ επιβάλλεται μία δομή στα δεδομένα, για να μην έχουμε απλά ανεξάρτητες τιμές από διαφορετικές Διωνυμικές κατανομές $y_i \sim b(n_i, p_i)$ με $\mu_i = n_i p_i$ και θα εκτιμάται από το $\tilde{\mu}_i = y_i$ [12].

Υπό την *H_S: το κορεσμένο μοντέλο* θα έχουμε ($\tilde{p}_i = \frac{y_i}{n_i}$):

$$\hat{l}_{iS} = ln\binom{n_i}{y_i} + y_i ln\tilde{p}_i + (n_i - y_i)\ln(1 - \tilde{p}_i)$$

Υπό την *H_0: το υποψήφιο μοντέλο* θα έχουμε ($\hat{p}_i = \frac{\hat{\mu}_i}{n_i}$):

$$\hat{l}_{i0} = ln\binom{n_i}{y_i} + y_i ln\hat{p}_i + (n_i - y_i)\ln(1 - \hat{p}_i)$$

Η διαφορά τους:

$$\hat{l}_{i0} - \hat{l}_{iS} = y_i ln\left(\frac{\hat{\mu}_i}{y_i}\right) + (n_i - y_i)\ln(\frac{n_i - \hat{\mu}_i}{n_i - y_i})$$

Η τυποποιημένη ελεγχοσυνάρτηση Deviance ορίζεται ως:

$$D(\widehat{\boldsymbol{\beta}}) = -2(\hat{l}_0 - \hat{l}_S) = 2\sum_{i=1}^{n}\{y_i ln\left(\frac{y_i}{\hat{\mu}_i}\right) + (n_i - y_i)\ln\left(\frac{n_i - y_i}{n_i - \hat{\mu}_i}\right)\} = 2\sum_{i=1}^{n} d_i(\widehat{\boldsymbol{\beta}}) \quad (2.9)$$

Αυτή η συνάρτηση είναι πολύ χρήσιμη διότι συγκρίνει τις παρατηρήσεις y_i και τα εκτιμημένα $\hat{\mu}_i$. Έτσι τώρα, μπορούμε να συγκρίνουμε τα δύο μοντέλα χρησιμοποιώντας την κατανομή X^2 ασυμπτωτικά. Για μεγάλες τιμές της διαφοράς $D_0 - D_1$, απορρίπτουμε την *H_0.* Εννοείται ότι η διαφορά είναι πάντα θετική, διότι το κορεσμένο (εμπεριέχει όλες τις μεταβλητές) έχει περισσότερες μεταβλητές από το υποψήφιο μοντέλο. Οπότε για να επιλέξουμε το βέλτιστο μοντέλο με τον πιο πάνω έλεγχο, αρκεί να επιλέξουμε το μοντέλο με τη μεγαλύτερη απόκλιση (Deviance).

Μικρή παρατήρηση είναι το γεγονός ότι για δυαδικά δεδομένα, δηλαδή η απόκριση $\boldsymbol{y}$ να παίρνει τιμές 0 και 1, με την ελεγχοσυνάρτηση Deviance δε μπορούμε να αποφανθούμε για την καταλληλότητα ενός μοντέλου. Όμως η συνάρτησή της χρησιμοποιείται και σαν εναλλακτική περίπτωση στο κριτήριο BIC για την επιλογή κατάλληλου μοντέλου. Οπότε δε μπορούμε να πούμε ότι δε μας χρειάζεται αυτός ο έλεγχος στα γενικευμένα γραμμικά μοντέλα.

2.2.7 Παράδειγμα Λογιστικής Παλινδρόμησης στην R

Στο παρακάτω παράδειγμα θα μελετήσουμε τη χρήση της λογιστικής παλινδρόμησης σε ένα μοντέλο για το αν κάποιος απόφοιτος Πανεπιστημίου στην Αμερική θα γίνει δεχτός για μεταπτυχιακό ή όχι. Οι ανεξάρτητες μεταβλητές μας είναι η βαθμολογία του στις εξετάσεις GRE, ο μέσος όρος του πτυχίου του (GPA) και το πόσο καλό είναι το πανεπιστήμιο από το οποίο αποφοίτησε (rank). Τα δεδομένα μας μπορούν να βρεθούν στο (*http://www.ats.ucla.edu/stat/data/binary.csv*) και είναι 401 δεδομένα με τις προαναφερθείσες ανεξάρτητες μεταβλητές και μεταβλητή απόκρισης 0 αν ο υποψήφιος έγινε δεκτός και 1 αν δεν έχει γίνει δεκτός (admit). Θεωρούμε τις μεταβλητές GPA και GRE συνεχείς και την μεταβλητή rank κατηγορική με τα ιδρύματα με βαθμό 1 να είναι τα καλύτερα.

Θα πάμε τώρα να αναλύσουμε τα δεδομένα μας με την λογιστική παλινδρόμηση. Πριν αρχίσουμε ορίζουμε τη μεταβλητή rank ως factor για να υπολογιστεί ως κατηγορηματική

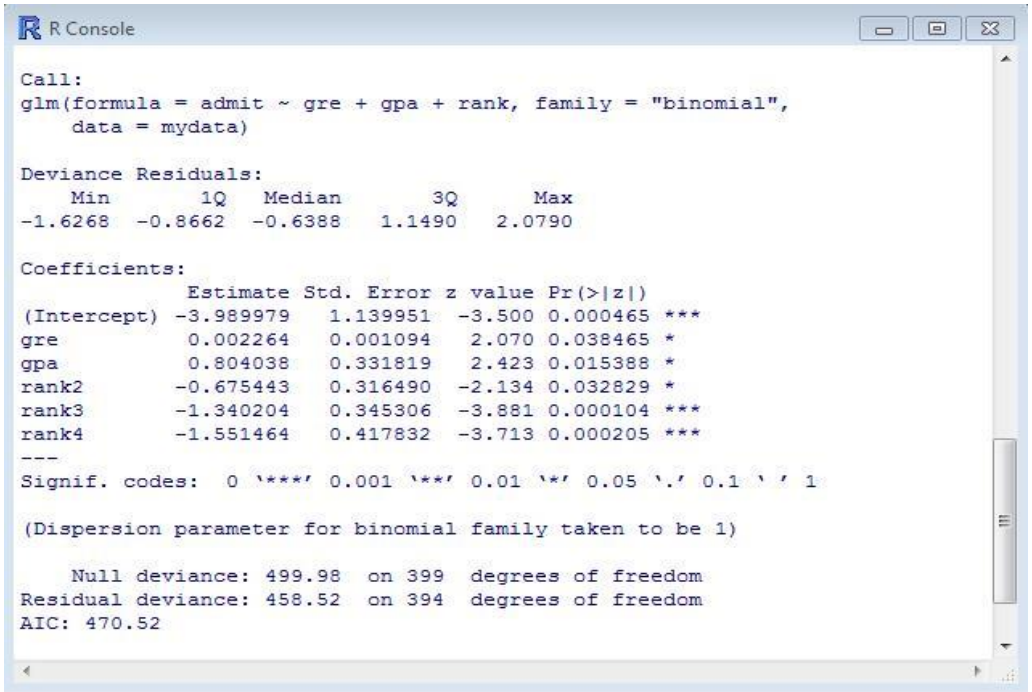
mydata$rank <- factor(mydata$rank)

Και ορίζουμε τη λογιστική παλινδρόμηση με τον ακόλουθο τύπο:

logistiki <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")

και για να δούμε τα αποτελέσματα πατάμε

summary(mylogit)



Εικόνα 7: Αποτελέσματα Λογιστικής παλινδρόμησης στην R

Στα αποτελέσματα μας βλέπουμε καταρχάς την null deviance και residual deviance στο τέλος που μας λέει πόσο καλά είναι προσαρμοσμένο το μοντέλο μας. Όπως είδαμε και πιο πριν η null deviance συμβολίζει την ελεγχοσυνάρτηση deviance για ένα μοντέλο με 0 παραμέτρους και ακολουθεί την X^2κατανομή με 399 βαθμούς ελευθερίας. Αντίστοιχα η residual deviance συμβολίζει την τιμή της ελεγχοσυνάρτηση μετά που προσθέσαμε τις παραμέτρους μας. Στη δεξιά στήλη των coefficients βλέπουμε την p-τιμή του έλεγχου wald για κάθε μεταβλητή μου ξεχωριστά. Επίσης οι τιμές στη στήλη estimate μας λένε πως για τα κάθε αύξηση κατά μία μονάδα της βαθμολογίας στα GRE οι λογαριθμημένες πιθανότητες αποδοχής αυξάνονται κατά 0.0022 και ανάλογα του GPA κατά 0.804. Τα estimates για το rank έχουν άλλη επεξήγηση που είναι η σύγκριση του πόσο μειώνεται η λογαριθμημένη πιθανότητα αν αντί να φοιτήσεις σε πανεπιστήμιο με rank 1 φοίτησες σε πανεπιστήμιο με rank 2,3 και 4 αντίστοιχα.

Συνεχίζοντας την ανάλυση του μοντέλου μας για να πάρουμε το Διάστημα Εμπιστοσύνης με βάση την πιθανοφάνεια πληκτρολογούμε *(confint(mylogit)*) και παίρνουμε

	2.5 %	97.5 %
(Intercept)	-6.2716202334	-1.792547080
gre	0.0001375921	0.004435874
gpa	0.1602959439	1.464142727
rank2	-1.3008888002	-0.056745722
rank3	-2.0276713127	-0.670372346
rank4	-2.4000265384	-0.753542605

Με τις εντολές wald.test(…) μπορούμε να κάνουμε του ελέγχους wald και να συγκρίνουμε μεταξύ πιθανών μοντέλων.

Κλείνοντας το παράδειγμά μας στην R θεωρώ πολύ σημαντικό να δούμε το πώς μπορούμε να κάνουμε προβλέψεις για καινούργιες παρατηρήσεις με βάση το μοντέλο που δημιουργήσαμε. Αυτός άλλωστε είναι και ο κύριος σκοπός κάθε στατιστικής ανάλυσης (όχι πάντα αλλά επί το πλείστον).

Για να κάνουμε προβλέψεις δημιουργούμε ένα νέο data set με την εντολή:

newdata1 <- with(mydata, data.frame(gre = mean(gre), gpa = mean(gpa), rank = factor(1:4)))
όπου όλες οι τιμές για το gre και gpa παίρνουν τις τιμές του μέσου όρου και το rank μεταβάλλεται από το 1 μέχρι το 4. Για να κάνουμε την πρόβλεψη χρησιμοποιούμε την εντολή:

newdata1$rankP <- predict(mylogit, newdata = newdata1, type = "response")

που δημιουργεί ένα data frame και προβλέπει τα αποτελέσματα βάση του mylogit (του μοντέλου λογιστικής παλινδρόμησης που προβλέψαμε). Παίρνουμε τα ακόλουθα αποτελέσματα:

	gre	gpa	rank	rankP
1	587.7	3.3899	1	0.5166016
2	587.7	3.3899	2	0.3522846
3	587.7	3.3899	3	0.2186120
4	587.7	3.3899	4	0.1846684

Άρα βάση των αποτελεσμάτων κάποιος απόφοιτος πανεπιστημίου με κατάταξη 1 έχει 51% πιθανότητα να γίνει δεκτός σε αντίθεση με κάποιον που φοίτησε σε πανεπιστήμιο κατάταξης 2 και έχει 35% αντίστοιχα.

2.3 Δέντρα αποφάσεων

Τα δέντρα αποφάσεων είναι ισχυρά εργαλεία της μορφής «If-Then» που χρησιμοποιούνται σε πολλούς τομείς όπως η στατιστική, πληροφορική, επιχειρησιακή έρευνα κτλ. Ο στόχος τους είναι ξεκινώντας από ένα σετ δεδομένων με παραμέτρους να ταξινομήσουν ή να προβλέψουν τη μεταβλητή απόκρισης βάση των παραμέτρων χρησιμοποιώντας ένα δενδροδιάγραμμα.

Η διαδικασία που ακολουθείται από τα δέντρα αποφάσεων είναι η εξής: Ξεκινώντας από την ρίζα του δέντρου όπου και έχουμε όλα τα δεδομένα του σετ εκπαίδευσης μας, διασπούμε σε κάθε κόμβο (βήμα) τα δεδομένα με βάση την βέλτιστη ιδιότητα του κόμβου που υπολογίζεται με κάποια κριτήρια όπως Gini index, gain ratio, information gain. Έτσι σε κάθε επόμενο κόμβο αποκτούμε συνεχώς και μεγαλύτερη ομοιογένεια καταλήγοντας τελικά στα φύλλα που περιέχουν τις κλάσεις του προβλήματος.

Στη συνέχεια και αφού έχουμε δημιουργήσει το δέντρο απόφασης μας ελέγχεται η απόδοση του δέντρου μας στο σετ δοκιμής. Όπως και με τις άλλες τεχνικές που περιγράψαμε, στα προηγούμενα μοντέλα δίνουμε ως είσοδο στο δέντρο τις τιμές του διανύσματος εισόδου και περιμένουμε ως απάντηση την τάξη του. Το πλήθος των σωστών αποφάσεων καθορίζει την ακρίβεια του δέντρου.

Τα δέντρα απόφασης έχουν 3 σημαντικά χαρακτηριστικά:

1. Αν όλα τα αντικείμενα είναι διαχωρίσιμα μεταξύ τους, δηλαδή δεν υπάρχουν όμοια αντικείμενα με διαφορετική τάξη, τότε μπορούμε να κατασκευάσουμε ένα δέντρο απόφασης με μηδενικό σφάλμα. Αυτό το γεγονός θέτει τα δέντρα αποφάσεων στην κατηγορία των ασταθών ταξινομητών αφού μια μικρή αλλαγή στα δεδομένα εκπαίδευσης οδηγεί σε ένα εντελώς διαφορετικά δομημένο δέντρο απόφασης.
2. Τα δέντρα ταξινομητές είναι διαισθητικά γιατί η διαδικασία απόφασης μπορεί να ακολουθηθεί αντίστροφα σαν μια σειρά απλών αποφάσεων. Οι δενδροειδής δομές μπορούν να συλλάβουν τη γνώση σε ιεραρχική δομή, με πιο διάσημα παραδείγματα την βοτανολογία, ζωολογία και ιατρικές διαγνώσεις.
3. Τόσο τα ποσοτικά όσο και τα ποιοτικά χαρακτηριστικά είναι κατάλληλα για το κτίσιμο δενδροειδών ταξινομητών. Ειδικά τα binary χαρακτηριστικά και χαρακτηριστικά με μικρό αριθμό κλάσεων είναι πολύ χρήσιμα γιατί η απόφαση μπορεί να διακλαδωθεί με ευκολία. Για ποσοτικές μεταβλητές ένα σημείο διαχωρισμού πρέπει να βρεθεί για να διαμορφωθεί η μεταβλητή σε κατηγορική. Για αυτό το λόγο και τα δέντρα αποφάσεων θεωρούνται *μη αριθμητικές μέθοδοι* για ταξινόμηση.

Δυαδικοί vs μη δυαδικοί διαχωρισμοί

Για να αυτοματοποιήσουμε την κατασκευή του δέντρων είναι λογικό να διαλέξουμε binary δέντρα, έτσι ώστε κάθε μη τερματικός κόμβος να παράγει ακριβώς δύο κόμβους. Για κάθε κόμβο με πολλαπλές απαντήσεις υπάρχει αντίστοιχη binary απεικόνιση. Αντίστοιχα για συνεχείς μεταβλητές θέτουμε κάποια τιμή κατώφλι που χωρίζει πάλι τα δεδομένα μας σε 2 κόμβους. Για τακτικές κατηγορικές μεταβλητές με M διαδοχικές κατηγορίες υπάρχουν M-1 πιθανοί binary διαχωρισμοί. Αντίστοιχα για ονομάστηκες κατηγορικές μεταβλητές με M πιθανές τιμές υπάρχουν 2^{M-1} πιθανοί διαχωρισμοί.

2.3.1 Επιλογή χαρακτηριστικού για ένα κόμβο

Ο βασικός στόχος στο σχεδιασμό δενδροειδών ταξινομητών είναι η ακρίβεια και η απλότητα. Η κατασκευή δέντρου μπορεί να συνεχιστεί μέχρι να επιτύχουμε απόλυτη ομοιογένεια ή μέχρι επίτευξη επαρκής ομοιογένειας.

Έστω ένα πρόβλημα c κλάσεων, με Ω = $\{\omega_1, \omega_2, \ldots\ldots, \omega_c\}$ και P_j η πιθανότητα της τάξης ω_j σε κάποιο συγκεκριμένο κόμβο t του δέντρου μας. Αυτές οι πιθανότητες υπολογίζονται ως το ποσοστό σημείων από την αντίστοιχη κλάση στο σετ. Η ανομοιογένεια των κλάσεων μπορεί να μετρηθεί με πολλούς τρόπους:

- **Εντροπία:** $i(t) = -\sum_{j=1}^{c} \mathrm{Pj}\log \mathrm{Pj}$ με 0log0=0. Για μια ομοιογενή περιοχή i(t)=0 που είναι και η ελάχιστη τιμή της ανομοιογένειας. Αντίστοιχα στην περίπτωση που οι κλάσεις έχουν ομοιόμορφη κατανομή η ανομοιογένεια i(t)=log(c)
- **Δείκτης Gini:** $i(t) = 1 - \sum_{j=1}^{c} \mathrm{Pj}$ με i(t) = 0 για την ομοιογενή περιοχή και i(t)=(c-1)/c για την πιο ανομοιογενή.
- **Misclassification:** $(\tau) = 1 - \max_{j=1,2,..,c} \mathrm{Pj}$. Εδώ η ανομοιογένεια συσχετίζεται με την ακρίβεια ταξινόμησης. Δίνει το αναμενόμενο λάθος αν ο κόμβος αντικαθιστούταν από ένα φύλλο και η επιλεγμένη τάξη ήταν αυτή με το μεγαλύτερο P_j

Έστω ότι χωρίζουμε τον κόμβο t σε δύο θυγατρικούς κόμβους t_o και t_1 βασισμένη στο χαρακτηριστικό X που όπως δείξαμε πιο πάνω μπορεί χωρίς βλάβη της γενικότητας να παρθεί ως binary. Το κέρδος του διαχωρισμού t βρίσκεται στην γενική μείωση ανομοιογένειας Δi(t)

$$\Delta i(t) = i(t) - P(X = 0)x_i(t_0) - P(X = 1)x_i(t_1) =$$

$$i(t) - P(X = 0)x_i(t_0) - \big(1 - P(X = 0)\big)x_i(t_1)$$

2.3.2 Κριτήρια διακοπής

Η κατασκευή του δέντρου μπορεί να συνεχιστεί εωσότου δεν υπάρχουν επιπλέον ανομοιογενής κόμβοι. Ειδικά στην περίπτωση μου το σετ εκπαίδευσης μας δεν περιέχει όμοια στοιχεία με διαφορετική κλάση είναι εφικτή η κατασκευή του τέλειου δέντρου . Όμως ένα τέτοιο δέντρο μπορεί τελικά να απομνημονεύει απλώς τις μεταβλητές

(overfitting) χωρίς να μπορεί να δώσει έγκυρα αποτελέσματα σε νέα σετ δεδομένων άρα δεν είναι χρήσιμο. Μία λύση για αυτό το πρόβλημα είναι να διακόψουμε τη διαδικασία πριν φτάσει σε ομοιογενής φύλλα. Πως μπορούμε όμως να αποφασίσουμε αν σταματήσαμε τη διαδικασία πολύ γρήγορα? Οι επιλογές μας για την διακοπή της διαδικασίας είναι οι παρακάτω έτσι ώστε να μη σταματήσουμε την διαδικασία πιο σύντομα από ότι χρειάζεται:

- Χρησιμοποίηση ενός σετ επικύρωσης από το σετ εκπαίδευσης. Όταν το σφάλμα στο σετ επικύρωσης ξεκινήσει να αυξάνετε σταμάτα το διαχωρισμό.
- Θέσε μια μικρή τιμή ως *κατώφλι μείωσης ομοιογένειας* = β. Όταν η μικρότερη πιθανή μείωση ομοιογένειας σε ένα κόμβο t είναι Δi(t) <β, σταμάτα το διαχωρισμό και ταξινόμησε τον κόμβο όπως την πλειοψηφία των αποτελεσμάτων.
- Θέσε μια τιμή κατώφλι για τον αριθμό των πόντων σε ένα κόμβο.
- Ποινικοποίησε την πολυπλοκότητα ενός δέντρου χρησιμοποιώντας ένα κριτήριο όπως

  $$\text{Maximize } a \times s \sum_{\varphi\acute{\upsilon}\lambda\lambda\alpha\, t} i(t) \text{ με s=”μέγεθος”}$$

  Όπου το «μέγεθος» μπορεί να είναι ο συνολικός αριθμός κόμβων, κλαδιών ή φύλλων του δέντρου και a μία θετική σταθερά.
- Χρησιμοποίησε ένα έλεγχο υπόθεσης για τη διευκρίνιση αν ένας επιπλέον διαχωρισμός θα είναι προς όφελος σου ή όχι.

2.3.3 Κλάδεμα

Μερικές φορές η διακοπή ενός δέντρου απόφασης μπορεί να αποδειχτεί πολύ κοντόφθαλμη και να σταματήσει επικερδής διαχωρισμούς, ένα φαινόμενο που είναι γνωστό ως “*επίδραση ορίζοντα*”. Για να αντιμετωπίσουμε αυτό το φαινόμενο αφήνουμε το δέντρο να αναπτυχθεί πλήρως και στη συνέχεια το “*κλαδεύουμε*” σε ένα μικρότερο μέγεθος. Το κλάδεμα ψάχνει μια ισορροπία μεταξύ της αύξησης του σφάλματος εκπαίδευσης και της μείωσης του μεγέθους του δέντρου. Έτσι με τη μείωση του μεγέθους του δέντρου ελπίζουμε να μειώσουμε το overfitting. Τα κυριότερα κριτήρια κλαδέματος είναι τα εξής:

- **Κλάδεμα μειωμένου σφάλματος (Reduced Error Pruning):** Είναι η απλούστερη μέθοδος κλαδέματος. Χρησιμοποιούμε ένα επιπλέον σετ εκπαίδευσης, το “σετ

κλαδέματος", που δεν είναι φανερό στη φάση δημιουργίας του δέντρου. Ξεκινώντας από τα φύλλα και ανεβαίνοντας προς την ρίζα χρησιμοποιείται ένας απλός έλεγχος σφάλματος σε όλους τους κόμβους. Αντικαθιστούμε τον κόμβο με φύλλο και τον ταυτίζουμε με την πλειοψηφική κλάση. Στη συνέχεια υπολογίζουμε το σφάλμα του νέου δέντρου του σετ κλαδέματος. Αν το σφάλμα είναι μικρότερο από το συνολικό αντικαταστούμε τον κόμβο με φύλλο, αλλιώς κρατάμε ολόκληρο το υπόδεντρο του κόμβου. Αυτή η διαδικασία εγγυάται ότι το δέντρο που θα καταλήξουμε είναι το ελάχιστο δυνατό.

- **Κλάδεμα απαισιόδοξου σφάλματος (Pessimistic Error Pruning):** Αντίθετα με το REP σε αυτή τη μέθοδο το ίδιο σετ δεδομένων χρησιμοποιείται τόσο για την κατασκευή όσο και για το κούρεμα του δέντρου. Λόγω αυτής της διαφοράς δεν μπορούμε να χρησιμοποιήσουμε αμέσως το σφάλμα για το κλάδεμα και χρησιμοποιούμε μία προσέγγιση από τη ρίζα στα φύλλα. Έστω n ο αριθμός των δεδομένων μου που έφτασαν στον κόμβο t, και e(t) το πλήθος των σφαλμάτων πουν θα είχα αν ο t-ιστός κόμβος αντικαθιστούταν από φύλλο. Έστω τώρα T_t το υπόδεντρο του t, L_t το σετ φύλλων του T_t και e'(T_t) ο αριθμός σφαλμάτων του T_t με διόρθωση πολυπλοκότητας:

$$e'(T_t) = \sum_{l \in L_t} e(l) + \frac{|L_t|}{2}$$

Ο κόμβος t αντικαθίσταται από φύλλο αν:

$$e(t) \leq e'(T_t) + \sqrt{\frac{e'(T_t)[n - e'(T_t)]}{n}} - 1/2$$

Η πολυπλοκότητα του δέντρου εκφράζεται ως ο αριθμός των φύλλων του υπόδεντρου. Έτσι τελικά επέρχεται μία ισορροπία μεταξύ σφάλματος και πολυπλοκότητα, χωρίς όμως θεωρητική υποστήριξη για την καταλληλότητα αυτής της μεθόδου. Αυτή η μέθοδος μερικές φορές οδηγεί σε υπερβολικό ή μειωμένο κούρεμα.

- **Κλάδεμα Κρίσιμης Τιμής (Critical Value Pruning):** Σε αυτό το κριτήριο θέτουμε μία κρίσιμη τιμή ως κατώφλι. Το δέντρο ελέγχεται από κάτω προς τα πάνω (ξεκινώντας στου κόμβους αμέσως πριν τα φύλλα) και όλοι οι κόμβοι στους οποίους το σφάλμα είναι μικρότερο από την τιμή κατώφλι αντικαθιστούνται με φύλλα. Θέτοντας μια

αύξουσα ακολουθία ως κρίσιμες τιμές μπορούμε να σχεδιάσουμε μια οικογένεια δέντρων και να επιλέξουμε το δέντρο με το ελάχιστο σφάλμα στο σετ κλαδέματος.

- **Κλάδεμα κόστους πολυπλοκότητας (Cost complexity Pruning):** Μια από τις πιο ευρέως διαδεδομένες μεθόδους κουρέματος γνωστή και ως CART. Στο πρώτο βήμα μία παραμετρική οικογένεια δέντρων δημιουργείται, έστω T= $\{T_0,..,T_\kappa\}$. Το δέντρο T_{i+1} λαμβάνεται από το δέντρο T_i σύμφωνα με κάποιο κριτήριο συσχετισμένο με κάποια παραμετρική τιμή $\alpha_{i+1.}$ Για παράδειγμα το α μπορεί να είναι η αύξηση στο φαινομενικό σφάλμα ανά φύλλο:

$$\alpha = \frac{e(t) - \sum_{l \in L_t} e(l)}{n(|L_t| - 1)}$$

αυτό είναι το επιπρόσθετο σφάλμα αν αντικαταστήσουμε το t με φύλλο, προς το πλήθος των φύλλων που ξεφορτωθήκαμε. Αν το α είναι μικρότερο από την τιμή κατώφλι τότε ο κόμβος t αντικαθιστάται με φύλλο. Το δέντρο T_0 λαμβάνεται κλαδεύοντας το δέντρο για τα κλαδιά με α=0 και το Τκ το δέντρο ρίζα. Επιπλέον για κάθε i $\alpha_\iota<\alpha_{i+1}$. Έστω T_0 το αρχικό δέντρο. Υπολογίζουμε το α για το T_0 και θέτουμε α_1=min(α) . Το T_1 είναι το κουρεμένο δέντρο όπου οι κόμβοι με τιμή α αντικαταστήθηκαν με φύλλα. Συνεχίζουμε με τον ίδιο τρόπο έως ότου να έχουμε κατασκευάσει όλη την ακολουθία Τ και βρίσκουμε το καλύτερο δέντρο χρησιμοποιώντας ένα ανεξάρτητο σετ κουρέματος.

- **Κλάδεμα βασισμένο στο σφάλμα (Error-Based Pruning):** Αυτή η μέθοδος εισηγείται ότι οι κόμβοι μπορούν να αντικατασταθούν από τους θυγατρικούς μαζί με το εναπομείναν υπόδεντρο. Έτσι το δέντρο μπορεί να ανακατασκευαστεί κατά τη διάρκεια του κλαδέματος. Το κριτήριο που χρησιμοποιείται για τη λήψη απόφασης στον κόμβο t είναι βασισμένο στη στατιστική εκτίμηση του διαστήματος εμπιστοσύνης (ΔΕ) για το σφάλμα t. Το άνω σφάλμα του ΔΕ U_t υπολογίζεται χρησιμοποιώντας τα αντικείμενα που έφτασαν στον κόμβο t ως στατιστικά δείγματα και υποθέτοντας πως ο αριθμός των σφαλμάτων είναι Διωνυμική τυχαία μεταβλητή. Οι τιμές του U_t βρίσκονται και χρησιμοποιούνται για τον υπολογισμό του σφάλματος. Η απόφαση για τον κόμβο παίρνεται βάση του αναμενόμενου σφάλματος των θυγατρικών κόμβων.

Κεφάλαιο 3: Bagging και Boosting

Όπως έχουμε ήδη αναφέρει το πρόβλημα της ταξινόμησης είναι ένα από τα σημαντικότερα προβλήματα του data mining αφού θεωρείται ιδανικό για εξαγωγή και απόκτηση γνώσης. Ένα από τα κύρια θέματα που απασχολεί τους ερευνητές τα τελευταία χρόνια είναι η βελτίωση της ακρίβειας της ταξινόμησης που παίρνουμε από τους αλγορίθμους. Για αυτό το λόγο οι αλγόριθμοι του boosting και bagging που θα δούμε στη συνέχεια είναι στο επίκεντρο της προσοχής τα τελευταία χρόνια αφού έχουν ως στόχο τον συνδυασμό “αδύναμων” ταξινομητών (ταξινομητών που η ακρίβεια τους είναι λίγο καλύτερη από την τυχαία επιλογή) σε ένα “ισχυρό” ταξινομητή.

Το Boosting επιτυγχάνει την εύρεση αυτού του ισχυρού ταξινομητή συνδυάζοντας πολλούς διαδοχικούς αδύνατους ταξινομητές και επικεντρώνοντας την προσοχή του στα λάθη που έκαναν οι ταξινομητές μας στα προηγούμενα βήματα.

Το Bagging παρομοίως είναι ακόμα μια τεχνική που σχεδιάστηκε για την βελτίωση της απόδοσης των αλγορίθμων εκμάθησης μηχανής. Το Bagging συνδυάζει ένα μεγάλο αριθμό ταξινομητών, όπου κάθε ταξινομητής χρησιμοποιεί ένα bootstrap δείγμα του αρχικού σετ εκπαίδευσης.

Τόσο το Bagging όσο και το Boosting έχουν χρησιμοποιηθεί με επιτυχία στις μεθόδους εκμάθησης μηχανής για την βελτίωση αλγορίθμων ταξινόμησης όπως δέντρα αποφάσεων και νευρωνικά δίχτυα. Πρόσφατες μελέτες έχουν δείξει πως και οι δύο μέθοδοι δουλεύουν το ίδιο καλά και ίσως και λίγο καλύτερα στα νευρωνικά δίχτυα παρά στα δέντρα ταξινόμησης.

3.1 Bagging

Ο όρος Bagging εισήχθηκε από τον Breiman και δεν είναι τίποτα άλλο παρά ένα ακρώνυμο του όρου Bootstrap AGGregatING. Στόχος του Bagging είναι να βελτιώσει την ακρίβεια αλγορίθμων “αδύναμης” μάθησης σε αλγορίθμους ισχυρής μάθησης. Η ιδέα πίσω από το Bagging είναι τόσο απλή όσο και γοητευτική. Το σύνολο κατασκευάζεται από ταξινομητές που είναι bootstrap αντίγραφα του σετ εκπαίδευσης. Οι ταξινομητές μετά συνδυάζονται με “ψήφιση”, δηλαδή για κάθε διάνυσμα εισόδου ο κάθε ταξινομητής προβλέπει την μεταβλητή εξόδου και τελικά η τιμή με τις περισσότερους “ψήφους” επιλέγεται ως η μεταβλητή απόκρισης για το συγκεκριμένο διάνυσμα. Το Bagging έχει πολλά πλεονεκτήματα όπως την μείωση της συνδιακήμανσης και την αποφυγή του overfitting.

3.1.1Μαθηματικό μοντέλο

Πριν αρχίσουμε την ανάλυση του μοντέλου θα δώσουμε κάποιους ορισμούς:

- **Overfitting:** Το overfitting λαμβάνει χώρα όταν ένα στατιστικό μοντέλο αρχίζει να περιγράφει το τυχαίο λάθος (θόρυβο) αντί τη σχέση που γυρεύουμε. Το overfiiting γίνεται συνήθως στην περίπτωση που το μοντέλο μας είναι πού πολύπλοκο, δηλαδή έχει πολλές παραμέτρους σε σχέση με τον αριθμό των παρατηρήσεων. Στην πραγματικότητα το overfitting γίνεται όταν το μοντέλο αρχίζει να απομνημονεύει αποτελέσματα από το σετ εκπαίδευσης αντι να μαθαίνει από αυτά.
- **Δειγματοληψία με αντικατάσταση:** Από το σετ δεδομένων μου επιλέγουμε τυχαία στοιχεία, τα οποία μπορούν να ξαναεπιλεγούν στη συνέχεια (ουσιαστικά σκεφτείτε ότι έχουμε ένα μπολ με μπάλες πολλών χρωμάτων, παίρνουμε μια μπάλα, καταγράφουμε το χρώμα και την επανατοποθετούμε στο μπολ μας.) Δείγμα bootstrap λέγεται κάποιο δείγμα που επιλέγεται από το αρχικό ομοιόμορφα και με αντικατάσταση.

Ξεκινάμε τώρα την επεξήγηση του τι ακριβώς κάνει το bagging. Το σκεπτικό πίσω από το bagging όπως είδαμε είναι η χρησιμοποίηση πολλών σετ εκπαίδευσης. Ιδανικά, αυτά τα

σετ εκπαίδευσης θα έπρεπε να παράγονται τυχαία από την κατανομή του προβλήματος. Στην πράξη όμως μπορούμε να έχουμε μόνο ένα σετ εκπαίδευσης , έστω

$$Z = \{ \mathbf{z}_1 , \mathbf{z}_2 , \ldots . \mathbf{z}_n \} \mu\varepsilon \ \mathbf{z}_j \ \epsilon \ R^n \ \gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ j = 1 \ldots . n$$

Από αυτό το σετ εκπαίδευσης θα παράγονται όλα τα υπόλοιπα σετ. Από το Z δημιουργούμε L τυχαία σετ εκπαίδευσης Z_i . Αυτό το πετυχαίνουμε επιλέγοντας ομοιόμορφα και με αντικατάσταση (bootstrap δείγμα) από το αρχικό σετ Z. Το μέγεθος κάθε Z_i είναι m$\leq n$. (Αν m=n τότε μόνο τα $(1 - \frac{1}{e})\approx$ 63,2% στοιχεία θα είναι διαφορετικά μεταξύ τους και τα υπόλοιπα θα είναι αντίγραφα [3]). Για να χρησιμοποιήσουμε τις παραλλαγές μέσα στο Z ο ταξινομητής βάσης πρέπει να είναι ασταθής. Αυτό πρέπει να γίνεται έτσι ώστε οι μικρές αλλαγές στο σετ εκπαίδευσης να οδηγούν σε μεγάλες αλλαγές στην έξοδο του ταξινομητή. Αν αυτό δεν συμβαίνει το σύνολο θα αποτελείται από μια συλλογή σχεδόν πανομοιότυπων ταξινομητών. Παραδείγματα ασταθών ταξινομητών είναι τα νευρωνικά δίκτυα και τα δέντρα αποφάσεων. Αξίζει να σημειωθεί ότι ο αλγόριθμος του Bagging τρέχει παράλληλα με τον ταξινομητή.

3.1.2 Αλγόριθμος εκπαίδευσης

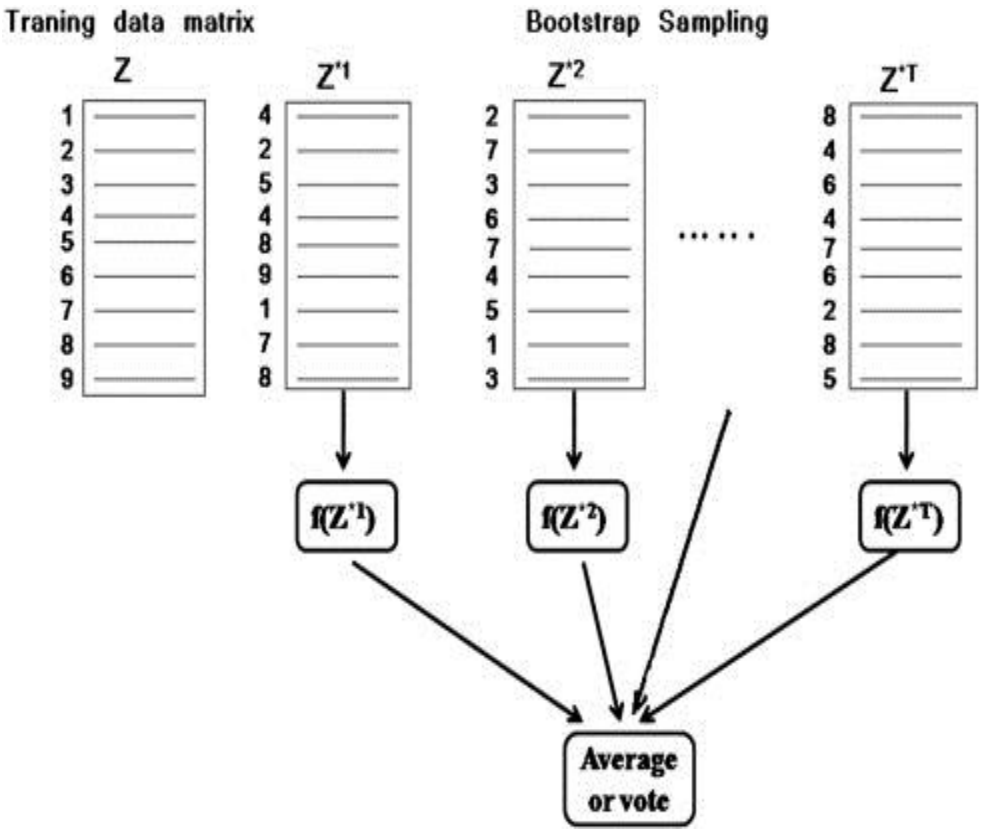
Φάση εκπαίδευσης

1) Αρχικοποίηση παραμέτρων
    - D = 0 το σύνολο των ταξινομητών
    - L = 0, ο αριθμός των ταξινομητών που θα εκπαιδεύσει
2) Για k = 1,2,…,L
    - Παίρνουμε ένα bootstrap δείγμα από το S_k από το Z
    - Εκπαιδεύουμε ένα ταξινομητή D_k χρησιμοποιώντας το σετ εκπαίδευσης S_k
    - Προσθέτουμε τον ταξινομητή στο σύνολο D = D ∪ D_k
3) Επέστρεψε το D.

Φάση ταξινόμησης:

4) Τρέξε D_1 , D_2, …. D_L για την είσοδο **x.**
5) Η κλάση με τις περισσότερες ψήφους επιλέγεται ως η κλάση απόκρισης για το διάνυσμα **x**

Αν οι έξοδοι των ταξινομητών ήταν ανεξάρτητες και οι ταξινομητές είχαν την ίδια ακρίβεια p, τότε η μέθοδος ψήφισης εγγυημένα θα βελτιώνει την απόδοση του μοντέλου μας. Το Bagging προσπαθεί να πετύχει την ανεξαρτησία των εξόδων με τον αν παίρνει bootstrap αντίτυπα από το σετ εκπαίδευσης. Τα δείγματα είναι ψευδοανεξάρτητα αφού προέρχονται από το ίδιο Z. Από την άλλη όμως ακόμα και αν ήταν ανεξάρτητα οι έξοδοι τους μπορεί να μην ήταν ανεξάρτητες.



<u>Σχήμα 9</u>: Σχηματική απεικόνιση της Bagging

3.1.3 Random Forest

Η μέθοδος Random Forest είναι μια τεχνική εκμάθησης για ταξινόμηση που δημιουργήθηκε από τον Breimann και στην πράξη κατασκευάζει μια πληθώρα δέντρων αποφάσεων και έχει ως απόκριση την κλάση που παρουσιάζεται συχνότερα ως απόκριση των επί μέρους δέντρων αποφάσεων. Ο ακριβής ορισμός του είναι:

Ορισμός: Ένα Random Forest είναι ένας ταξινομητής που αποτελείται από μια συλλογή ταξινομητών μορφής δέντρων αποφάσεων. Κάθε δέντρο απόφασης αναπτύσσεται σε σχέση με ένα τυχαίο διάνυσμα $\boldsymbol{\Theta}_{\kappa}$ όπου $\boldsymbol{\Theta}_{\kappa}$ κ=1….L είναι ανεξάρτητα και ισόνομα μεταξύ τους. Κάθε δέντρο "ψηφίζει" για την πιο δημοφιλή κλάση εισόδου **x.**

Άρα ένα random forest μπορεί να κτιστεί με δειγματοληψία από το σετ χαρακτηριστικών, από το σετ δεδομένων ή απλώς από τυχαία μεταβολή παραμέτρων του δέντρου.

Ένα από τα πιο επιτυχημένα ευρήματα του random forest είναι η τυχαία επιλογή εισόδου (input). Ενώ επιλέγουμε τυχαία bootstrap δείγματα από το σύνολο των δεδομένων εκπαίδευσης, η τυχαία επιλογή λαμβάνει χώρα σε κάθε κόμβο του δέντρου. Διαλέγουμε τυχαία ένα subset S με M χαρακτηριστικά από το αρχικό σετ ν χαρακτηριστικών και ψάχνουμε από το S το καλύτερο χαρακτηριστικό για να διαχωρίσουμε τον κόμβο. Ο Breiman εισηγείται τη δημιουργία ενός ολόκληρου CART δέντρου αποφάσεων με αυτόν τον τρόπο. Η εισηγούμενη τιμή για το M είναι $[\log_2 n + 1]$ όπου n ο συνολικός αριθμός χαρακτηριστικών.

3.1.3.1 Αλγόριθμος

Κάθε δέντρο κατασκευάζεται βάση του ακόλουθου αλγόριθμου:

1. Έστω N ο αριθμός των περιπτώσεων εκπαίδευσης και M ο αριθμός των μεταβλητών στον ταξινομητή.
2. Ο αριθμός m των μεταβλητών εισόδου χρησιμοποιείται για να παρθεί απόφαση σε ένα κόμβο του δέντρου. m << M
3. Επιλέγουμε ένα σετ εκπαίδευσης για το δέντρο διαλέγοντας n φορές με αντικατάσταση από όλες τις N διαθέσιμες περιπτώσεις εκπαίδευσης (bootstrap

δείγμα). Χρησιμοποιώντας τις υπόλοιπες υποθέσεις υπολογίζουμε το σφάλμα του δέντρου, προβλέποντας τις κλάσεις του.

4. Για κάθε κόμβο στο δέντρο επιλέγουμε τυχαία m μεταβλητές στις οποίες στηρίζουμε την απόφαση του κόμβου. Υπολογίζουμε τον καλύτερο δυνατό διαχωρισμό βασισμένο στις m μεταβλητές του σετ εκπαίδευσης.
5. Κάθε δέντρο αναπτύσσεται πλήρως χωρίς κλάδεμα.

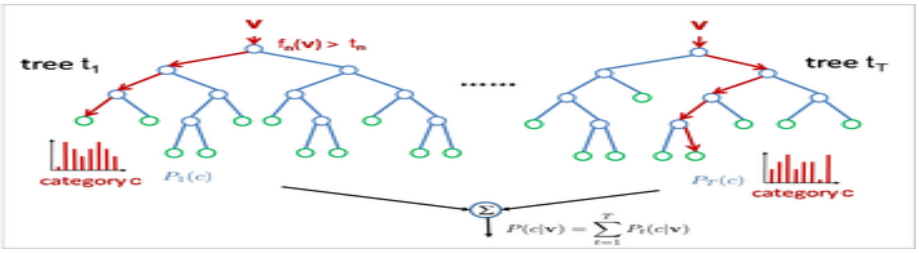
3.1.3.2 Σημαντικότητα μεταβλητών

Όπως θα δούμε και στην εφαρμογή μας στη συνέχεια τα Random Forests χρησιμοποιούνται για βαθμολόγηση της σημαντικότητας των μεταβλητών στα προβλήματα ταξινόμησης με φυσικό τρόπο. Η ακόλουθη τεχνική περιγράφηκε από τον Breimann και υπάρχει στο στατιστικό πακέτο R.

Το πρώτο βήμα για να μετρήσουμε τη σημαντικότητα κάποιας μεταβλητής σε ένα σετ δεδομένων $D_n = \{(X_i, Y_i)\}_{i=1}^{n}$ είναι να προσαρμόσουμε ένα random Forest στα δεδομένα μας. Κατά τη διάρκεια της προσαρμογής το σφάλμα κάθε δεδομένου καταγράφεται και υπολογίζεται ο μέσος όρος του στο δάσος.

Για να μετρήσουμε τη σημαντικότητα του i-οστού χαρακτηριστικού μετά την εκπαίδευση, οι τιμές του i-οστού χαρακτηριστικού μετατίθενται μεταξύ των δεδομένων εκπαίδευσης και το σφάλμα επαναϋπολογίζεται στο μετατιθέμενο σετ δεδομένων. Η σημαντικότητα του i-οστού χαρακτηριστικού υπολογίζεται βρίσκοντας τον μέσο όρο της διαφοράς του σφάλματος πριν και μετά την μετάθεση όλων των δέντρων. Η βαθμολογία κανονικοποιείται με την τυπική απόκλιση αυτών των διαφορών.

Λογικά τα χαρακτηριστικά που παράγουν μεγάλες τιμές ταξινομούνται ως πιο σημαντικά από τα χαρακτηριστικά με μικρότερες τιμές.



<u>Εικόνα 8</u> : Γραφική απεικόνιση των Random Forests

3.2 Boosting

Το boosting σχετίζεται με το " γενικό πρόβλημα παραγωγής ενός πολύ ακριβούς μοντέλου συνδυάζοντας μερικώς ακριβείς γενικούς κανόνες". Στόχος των αλγορίθμων boosting είναι να μετατρέψουν "αδύναμους" ταξινομητές σε ένα ισχυρό ταξινομητή. Πλέον ο αλγόριθμος ADABoost (των Freund και Schaphire) είναι σχεδόν συνώνυμος με το boosting.

Πριν προχωρήσουμε σε περεταίρω ανάλυση και επεξήγηση των όρων που χρησιμοποιήσαμε θα επιχειρήσουμε να αναλύσουμε τι κάνει το bagging χρησιμοποιώντας ένα απλό παράδειγμα, έτσι ώστε να μπορέσουμε να καταλάβουμε το σκεπτικό πίσω από αυτό και να αντιληφθούμε γιατί βελτιώνει τους αλγόριθμούς μας.

3.2.1 Εισαγωγικό παράδειγμα

Κάποιος τζογαδόρος που στοιχηματίζει στον Ιππόδρομο αποφασίζει να δημιουργήσει ένα αλγόριθμο, ένα πρόγραμμα στον ηλεκτρονικό υπολογιστή, για να μεγιστοποιήσει τα κέρδη του. Αυτό το πρόγραμμα θα δέχεται ένα διάνυσμα με πληροφορίες για κάθε ιπποδρομία (ηλικία αλόγων, κατάταξη καβαλάρη, αποδόσεις κτλ) και θα προβλέπει τον νικητή κάθε ιπποδρομίας. Για να δημιουργήσει το πρόγραμμα λοιπόν πάει σε κάποιον ειδικό ιπποδρομιών που φυσικά αδυνατεί να του πει κάποιο γενικό κανόνα για το άλογο που θα κερδίζει πάντα. Αντιθέτως αν του παρουσιαστούν τα δεδομένα μιας ιπποδρομίας μπορεί με ευκολία να σου πει κάποιον επιμέρους κανόνα (rule of thumb) για την ιπποδρομία (π.χ. στοιχημάτισε στο άλογο με την καλύτερη απόδοση, με τις περισσότερες πρόσφατες νίκες κτλ.)

Αν και αυτοί οι επιμέρους κανόνες είναι λίγο άχαροι και ανακριβείς προφανώς μας δίνουν κάποια πρόβλεψη έστω και λίγο καλύτερη από την τυχαία επιλογή. Ακόμα, αν συνεχίσουμε να ρωτάμε τον ειδικό για περισσότερες ιπποδρομίες θα μπορεί να μας λέει όλο και περισσότερους επιμέρους κανόνες.

Άρα τώρα ο τζογαδόρος μας έχει κάποια μέθοδο για να βρίσκει επιμέρους κανόνες. Σε αυτό το σημείο παρουσιάζοντα 2 προβλήματα. Καταρχάς πως θα επιλέξει τις κούρσες που θα δείξει στον ειδικό για να πάρει τους επιμέρους κανόνες και δεύτερον μόλις έχει συλλέξει αρκετούς επιμέρους κανόνες πως θα τους συνδυάσει σε ένα κανόνα υψηλής ακρίβειας.

Αυτός ακριβώς είναι ο ρόλος του boosting! Είναι μια γενική και αποδεδειγμένα αποτελεσματική μέθοδος παραγωγής ενός πολύ ακριβούς κανόνα πρόβλεψης, που δημιουργείται από το συνδυασμό των σχετικά ανακριβών αυτών επιμέρους κανόνων.

3.2.2 Προέλευση του Boosting: Αλγόριθμος Hedge (β)

Το Boosting εμπνεύστηκε από ένα αλγόριθμο ονόματι Hedge (β). Αυτός ο αλγόριθμος κατανέμει τα βάρη σε ένα σετ στρατηγικών που προβλέπουν το αποτέλεσμα κάποιου συμβάντος. Το βάρος κάθε στρατηγικής s_i μπορεί να ερμηνευθεί ως η πιθανότητα το s_iνα είναι η καλύτερη στρατηγική πρόβλεψης από όλες(έστω τα βάρη μου w_i $τότε$ $\sum w_i = 1$). Η κατανομή των βαρών ενημερώνεται μετά από κάθε αποτέλεσμα έτσι ώστε οι στρατηγικές με σωστές προβλέψεις να λαμβάνουν περισσότερο βάρος ενώ τα βάρη των στρατηγικών με βάρος στο προηγούμενο βήμα να ελαττώνονται.

Ας συσχετίσουμε τώρα το σκεπτικό του Hedge(β) στον συνδυασμό ταξινομητών που έχουμε δημιουργήσει με το bagging. Έχουμε ότι οι ταξινομητές αντιπροσωπεύουν τις στρατηγικές και το «γεγονός» είναι η χαρτογράφηση μιας τυχαίας z_jαπό το Z. έστω $\{D_1, D_2, \dots, D_K\}$ το σύνολο των ταξινομητών μας. Προφανώς δεν ξέρουμε ποιος από αητούς τους ταξινομητές έχει την καλύτερη ακρίβεια στο πρόβλημά μας. Παίρνουμε ένα τυχαίο σημείο **z** από το Z (σετ εκπαίδευσης) και έχουμε κάθε ταξινομητή $D_ι$να κάνει μια πρόβλεψη για το αποτέλεσμά του.

Ορίζουμε την “απώλεια” ενός τυχαίου ταξινομητή να είναι 1 αν ο ταξινομητής είναι λάθος και 0 αν είναι σωστός. Η αναμενόμενη απώλεια από τον τυχαίο μας ταξινομητή είναι ο μέσος όρος των ταξινομητών του D που χαρτογράφησαν λάθος το σημείο μας. Από τη στιγμή που θέλουμε να αυξήσουμε την ακρίβεια πρόβλεψης για το επόμενο σημείο αυξάνουμε την πιθανότητα επιλογής κάποιου D_iαπό το D που είχε περισσότερες επιτυχίες στα προηγούμενα σημεία. Άρα αλλάζουμε την κατανομή στο D καθώς ταξινομούμε περισσότερα παραδείγματα.

Έστω ότι η πρόβλεψη που κάνουμε βασίζεται στην τυχαία επιλογή κάποιου ταξινομητή από το D, σύμφωνα με την πιθανότητα από την πιθανότητα που δίνεται από την κατανομή στο D. Ο Hedge (β) εξελίσσει την κατανομή στο D για να ελαχιστοποιήσει το συσσωρευτικό λάθος της πρόβλεψης. Η κατανομή αυτή υπολογίζεται κανονικοποιώντας τα βάρη που

ενημερώνονται μετά από κάθε επιλογή του z_j. Ο Freund και Schaphire αποδεικνύουν ότι το άνω όριο της απώλειας δεν είναι πολύ χειρότερο από αυτό του καλύτερου ταξινομητή στον D (το οποίο και δεν ξέρουμε).

Γενικότερα υποθέτουμε ότι η απώλεια l_i^j παίρνει τιμές στο διάστημα [0,1] αντί μόνο τις διακριτές τιμές {0,1}. Άρα έχουμε μια συνεχή λ_i ϵ [0,1] και Λ ϵ [0, L N]. Ο Freund και Schaphire δείχνουν ότι αν επιλέξουμε

$$\beta = g(\hat{\lambda}/\ln L) \ \mu\varepsilon\ g(\zeta) = {1}\Big/{1+\sqrt{\frac{2}{\zeta}}}$$

Με $\hat{\lambda}$ να είναι η εκτίμηση της απώλειας του καλύτερου ταξινομητή μας, τότε η συνολική απώλεια φράσσεται από

$$\Lambda \leq \min_{i=1\ \mu\varepsilon\chi\rho\iota\ L} \lambda_i + \sqrt{2\lambda\ lnL} + lnL$$

3.2.2.1 Αλγόριθμος Hedge(β)

Δεδομένα:

- D={ D_1 , D_2 , … . , D_k } :το σύνολο ταξινομητών
- Z= {$\boldsymbol{z}_1$, $\boldsymbol{z}_2$, . . . , $\boldsymbol{z}_n$ } τα δεδομένα εκπαίδευσης.

1) Υπολογισμός παραμέτρων
   - Διαλέγουμε β ϵ [0,1] ο πολλαπλασιαστής μας για ελάττωση των βαρών
   - Θέτουμε αρχικά βάρη $\boldsymbol{w}^1$= $(w_{1,}w_2,\ldots,w_n)$, $w_i^1 \in [0,1]$ $\sum_{i=1}^{L} w_i^1 = 1$ (συνήθως θέτουμε $w_i^1 = 1/L$)
   - Λ = 0 η συσσωρευτική απώλεια
   - $\lambda_{\boldsymbol{\iota}} = 0$ η απώλεια σε κάθε $i\ \mu\varepsilon\ 0 \leq \iota \leq L$
2) Για κάθε $z_j\ \mu\varepsilon\ j = 1 \ldots . . N$

- Υπολογίζουμε την κατανομή $D_{i=1}^j = {w_i^j}\Big/{\sum_{k=1}^{L} w_k^j}$ με i=1….L

- Βρίσκουμε τις L απώλειες ($\lambda_i^j = 1\ \alpha\nu\ D_i$ ταξινομεί λάθος το z_j και λ_i^j =0 αν D_i ταξινομεί σωστά το z_j j=1….L
- Ενημερώνουμε τη συσσωρευτική απώλεια

$$\Lambda \text{ <- } L + \sum_{i=1}^{L} p_i^j l_i^j$$

- Ενημερώνουμε τις απώλειες

$$\lambda_i < -\lambda_i + \ l_i^j$$

- Ενημερώνουμε τα βάρη

$$w_i^{j+1} < -w_i^j \beta^{l_i^j}$$

3) Υπολογισμός και επιστροφή των Λ, $\lambda_i, p_i^{n+1}\ \ i = 1 \dots L$

3.2.3 Adaboost

Το AdaBoost είναι ακρωνύμιο του ADAptive BOOSTing που δημιουργήθηκε από τους Schaphire και Freund. Αν και ο adaboost είναι ευαίσθητος στις ακραίες τιμές και το θόρυβο, σε πολλές περιπτώσεις είναι λιγότερο επιρρεπής στο overfitting από άλλους αλγόριθμους εκμάθησης. Οι ταξινομητές που χρησιμοποιεί μπορεί να είναι αδύνατοι αλλά όσο είναι καλύτεροι από την τυχαία επιλογή το τελικό μας μοντέλο θα είναι βελτιωμένο. Ακόμα και ταξινομητές με χειρότερη ακρίβεια από την τυχαία επιλογή μπορεί να αποδειχτούν χρήσιμοι αφού στο τέλος θα λάβουν αρνητικά βάρη και θα συμπεριφέρονται σαν τον αντίστροφό τους.

Η ιδέα πίσω από το Adaboost είναι να αναπτύξεις το σύνολο D σταδιακά , προσθέτοντας ένα ταξινομητή σε κάθε βήμα. Ο ταξινομητής που εισέρχεται στο κ βήμα εκπαιδεύεται σε επιλεγμένα δεδομένα από το σετ εκπαίδευσης Z. Η κατανομή δειγματοληψίας ξεκινά ως ομοιόμορφη και συνεχίζει αυξάνοντας την πιθανοφάνεια “δύσκολων” σημείων. Άρα η κατανομή ενημερώνεται κάθε φορά αυξάνοντας την πιθανοφάνεια αντικειμένων που ταξινομήθηκαν λάθος στο βήμα κ-1.

3.2.3.1 Αλγόριθμος AdaBoost

Φάση εκπαίδευσης

1) Αρχικοποίηση παραμέτρων

- Θέτουμε βάρη $\boldsymbol{w^1}$ = [$w_1, w_2, \dots w_n$] με $w_j^1 \in [0,1]$ $\sum_{i=1}^{L} w_i^1 = 1$ (συνήθως θέτουμε $w_i^1 = 1/L$)
- Θέτουμε το σύνολο ταξινόμησης D=0
- L ο αριθμός των ταξινομητών για εκπαίδευση.

2) Για κ= 1… L

- Παίρνουμε δείγμα S_k από το Z χρησιμοποιώντας την κατανομή w^k
- Κατασκευάζουμε ταξινομητή D_k χρησιμοποιώντας το S_k ως σετ εκπαίδευσης
- Υπολογίζουμε το συνολικό σφάλμα με βάρη στο κ βήμα

$$\varepsilon_\kappa = \sum_{j=1}^{n} w_j^k l_k^j$$

($l_k^j = 1$ αν ο D_kταξινομήσει λάθος το $\boldsymbol{z}_j$και l_k^j= 0 αλλιώς)

- Αν $\varepsilon_\kappa = 0$ ή $\varepsilon_\kappa \geq 0.5$ αγνόησε το D_k επανεκτιμήστε τα βάρη στο 1/N και συνέχισε.
- Αν όχι υπολόγισε $\beta_\kappa = {\varepsilon_\kappa}/{1 - \varepsilon_\kappa}$ $\mu\varepsilon$ $0 < \varepsilon_\kappa < 0.5$
- Ενημερώστε το ξεχωριστό βάρος

$$w_j^{k+1} = w_j^k \beta_k^{1-l_k^j} / \sum_{i=1}^{n} w_i^k \beta_k^{1-l_k^j} \quad \text{j=1…n}$$

3) Επέστρεψε το D , $\beta_1 \dots . \beta_L$

Φάση ταξινόμησης

4) Υπολόγισε την υποστήριξη για την κλάση ω_t με

$$\mu_t(\boldsymbol{x}) = \sum_{D_k(\boldsymbol{x})=\omega_t} \ln(\frac{1}{\beta_k})$$

5) Η κλάση με τη μέγιστη υποστήριξη επιλέγεται ως η κατάλληλη για το **x.**

Πλεονεκτήματα Adaboost

- Εύκολος, απλός και γρήγορος προγραμματισμός
- Η μοναδική παράμετρος που προσαρμόζεται είναι το πλήθος των επαναλήψεων T
- Δεν απαιτείται εκ των προτέρων γνώση του αδύναμου ταξινομητή άρα μπορεί αν εφαρμοστεί με οποιαδήποτε μέθοδοι αδύναμης ταξινόμησης
- Εντοπίζει ακραίες τιμές αφού επικεντρώνεται σε παρατηρήσεις οι οποίες ταξινομούνται δυσκολότερα. Άρα οι παρατηρήσεις με το υψηλότερο βάρος συχνά αποδεικνύεται ότι είναι ακραία σημεία.

Μειονέκτημα AdaBoost

- Λόγω του ότι ο AdaBoost δίνει τόση πολλή "προσοχή" στα σημεία που ταξινομούνται λάθος δεν είναι καθόλου ακριβής σε δεδομένα με θόρυβο.

3.2.3.2. Άνω όριο σφάλματος

Έχουν αποδειχθεί δύο σημαντικά θεωρήματα από τους Freund και Schaphire για το άνω όριο του σφάλματος του Adaboost.

Θεώρημα 3.2.3.2.1: Έστω Ω={ω_1, ω_2}, ε το συνολικό σφάλμα και ε_i , $i = 1 \ldots . L$ το σφάλμα με βάρη των ταξινομητών στο D, τότε:

$$\varepsilon > 2^L \prod_{i=1}^{L} \sqrt{\varepsilon_\iota (1 - \varepsilon_\iota)}$$

Στην περίπτωση που έχουμε πολλές πιθανές εξόδους τότε το θεώρημα γενικεύεται ως εξής.

Θεώρημα 3.2.3.2.2: Έστω Ω={$\omega_1, \omega_2 \ldots . \omega_L$}, ε το συνολικό σφάλμα και ε_i με $\varepsilon_i < 0.5 \;\; i = 1 \ldots . L$ το σφάλμα με βάρη των ταξινομητών στο D, τότε:

$$\varepsilon > 2^L \prod_{i=1}^{L} \sqrt{\varepsilon_\iota (1 - \varepsilon_\iota)}$$

3.2.3.3 Στοχαστικό Boosting

Το Boosting εκ φύσεως βασίζεται σε μία έρευνα φθίνουσας κλίσης για να βελτιστοποιήσει την συνάρτηση κόστους και να βρει τόσο τα βάρη όσο και την συνάρτηση εκμάθησης σε κάθε επανάληψη. Στην Στοχαστικό Boosting με κλίση (Stochastic Gradient Boosting), μια τυχαία μετατιθέμενη στρατηγική δειγματοληψίας εφαρμόζεται σε κάθε επανάληψη για να λάβει ένα αναθεωρημένο σετ εκπαίδευσης. Ο Αλγόριθμος του SGB είναι ο εξής.

1. **Αρχικοποίησε** την F(x):=0
2. **Για m=1 μέχρι M** :
   - Θέσε $w_i = -\frac{\delta L(y,g)}{\delta g}|_{g=f(x)}$
   - Μοντελοποίησε το y=η(h_m(x)) ως τον ταξινομητή βάσης σου με βάρη, χρησιμοποιώντας $|w_i|$ με το σετ εκπαίδευσης π_m.
   - Υπολόγισε το $\alpha_m = \arg\min_a \sum_{i\in\pi_m} L(y_i, F(x) + a\eta(h_m(x_i)))$
   - Ανανέωσε το $F(x) = F(x) + va_m\eta(h_m(x))$
3. **Τέλος**

Αυτός ο αλγόριθμος στη γενική του μορφή μπορεί να λειτουργήσει με αυθαίρετη συνάρτηση απώλειας, Στην εφαρμογή μας όμως και με τη συνάρτηση ada της R θα θέσουμε $L(y\ ,f)$=$\log(1+e^{-yf})$ (λογιστική) (η άλλη μας επιλογή είναι η εκθετική με L(y,f)= e^{-yf}) και ως η(x) =0.5 $\log(\frac{x}{1-x})$ (real); οι άλλες μας επιλογές είναι sign(x) και η(x)=x (gentle).

3.2.3.4 Real και Gentle Adaboost

Αν και εμπειρικά έχει δειχθεί ότι ο AdaBoost βελτιώνει την ακρίβεια πρόβλεψης, παράγει σε κάθε στάδιο ένα αντικείμενο με τιμή την προβλεπόμενη κλάση. Αυτή η "τραχιά" πληροφορία μικραίνει την αποτελεσματικότητα του αλγόριθμου στην εύρεση βέλτιστης λύσης. Για να υπερπηδήσουμε αυτή την δυσκολία προτείνονται πολλές λύσεις, μία εκ των οποίων είναι ο αλγόριθμος να παράγει μία αριθμητική τιμή σε κάθε βήμα αντί την κλάση του αντικειμένου. Αυτό κάνει ο Real Adaboost όπου η εκτιμημένη πιθανότητα τάξης μετασχηματίζεται χρησιμοποιώντας την $0.5\log(\frac{x}{1-x})$ σε μία κλίμακα πραγματικής τιμής. Αυτή η τιμή στη συνέχεια χρησιμοποιείται στην συνεισφορά της παρατήρησης στο τελικό

μοντέλο. Αυτή η τροποποίηση επιτρέπει στον Real Adaboost να βρει με μεγαλύτερη αποτελεσματικότητα την βέλτιστη ταξινόμηση. Επιπλέον, εκτός από την επέκταση του Real Adaboost ο Friedman εισηγείται ακόμα μία επέκταση, το Gentle Adaboost, που ελαχιστοποιεί την εκθετική συνάρτηση κόστους χρησιμοποιώντας βήματα Newton-Raphson (για την μέθοδο N-R παραπέμπουμε στο [40]).

Κεφάλαιο 4: Εφαρμογή

Πρόβλεψη κίνησης μετοχών στην R

Εισαγωγή:

Όπως αναφέραμε και στα προηγούμενα κεφάλαια, λόγω της πληθώρας δεδομένων που υπάρχουν για την κίνηση των μετοχών, το data mining φαίνεται να είναι η ιδανική επιλογή για την εξόρυξη πληροφοριών και την απόκτιση συγκριτικού πλεονεκτήματος σε σχέση με την ανθρώπινη επίβλεψη των δεδομένων μας. Από την άλλη ορισμένοι ερευνητές ισχυρίζονται πως ότι οι αγορές προσαρμόζονται τόσο γρήγορα στις αλλαγές της τιμής που δεν υπάρχει χρόνος για τη δημιουργία κέρδους με συνεχή τρόπο. Αυτή η θεωρία είναι γνωστή και ως «*υπόθεση επαρκούς αγοράς»,* που όμως στις μέρες μας έχει αντικατασταθεί με πιο «χαλαρές» υποθέσεις που αφήνουν περιθώρια για κέρδος..

Τα δεδομένα για την κίνηση των μετοχών όμως εμφανίζουν κάποιες ιδιαιτερότητες σε σχέση με δεδομένα σε άλλες εφαρμογές. Αυτές οι ιδιαιτερότητες πηγάζουν κυρίως από το γεγονός ότι οι μετοχές συμπεριφέρονται ως χρονοσειρές.

Στόχος μας σε αυτό το κεφάλαιο είναι να προσπεράσουμε αυτές τις δυσκολίες και να εφαρμόσουμε τις μεθόδους που αναλύσαμε στα 2 προηγούμενα κεφάλαια για την εξόρυξη πληροφοριών από τον δείκτη αγοράς (market index) S&P 500 (Standar & Poor 500). Τα δεδομένα διατίθενται ελεύθερα στο yahoo.com και κατεβαίνουν σε μορφή excel.

Αξίζει τέλος να σημειωθεί πως ο γενικός στόχος της ανάλυσης συναλλαγών μετοχών είναι να διαχειριστεί ένα χαρτοφυλάκιο αγοράζοντας και πουλώντας κεφάλαια. Αυτό ξεφεύγει από τους στόχους αυτής της εφαρμογής και της παρούσας εργασίας γενικότερα που είναι η εφαρμογή των μεθόδων του data mining στα χρηματοοικονομικά και η επίδειξη των αποτελεσμάτων αυτών των μεθόδων από στατιστική σκοπιά. Για αυτό το σκοπό αν και όπως θα δούμε στη συνέχεια τα αποτελέσματα μας παρουσιάζονται εξαιρετικά ενδιαφέροντα δεν αποτελούν σε καμία περίπτωση από μόνα τους επενδυτική στρατηγική.

4.1 Δεδομένα

Όπως αναφέραμε και στην εισαγωγή θα χρησιμοποιήσουμε δεδομένα του δείκτη αγοράς **S&P5000**. Κατεβάζοντας τα δεδομένα από το *yahoo.finance* βλέπουμε ότι περιέχουν την ημερομηνία (Date), τιμή ανοίγματος μετοχής (Open), τιμή κλεισίματος (Close), υψηλότερη και χαμηλότερη τιμή ημέρας (High, Low) και προσαρμοσμένη τιμή κλεισίματος (Adj.Close). τα δεδομένα μας είναι από το 1970 (2/1/1970) μέχρι το 2009 (15/9/2009) και είναι 10021 στο σύνολο.

head(GSPC)

	Open	High	Low	Close	Volume	Adjusted
1970-01-02	92.06	93.54	91.79	93.00	8050000	93.00
1970-01-05	93.00	94.25	92.53	93.46	11490000	93.46
1970-01-06	93.46	93.81	92.13	92.82	11460000	92.82
1970-01-07	92.82	93.38	91.93	92.63	10010000	92.63
1970-01-08	92.63	93.47	91.99	92.68	10670000	92.68
1970-01-09	92.68	93.25	91.82	92.40	9380000	92.40

Πίνακας 3: Μορφή Δεδομένων

4.1.1 Χειρισμός δεδομένων στην R

Τα δεδομένα μας για αυτή την εφαρμογή εξαρτώνται από τον χρόνο. Αυτό σημαίνει πως κάθε παρατήρηση έρχεται μαζί με την παράμετρο της ημερομηνίας παρατήρησής της. Λόγω αυτής της ιδιαιτερότητας η σειρά μεταξύ των παρατηρήσεων έχει σημασία (σε αντίθεση με άλλες εφαρμογές).

Γενικά η μορφή των παρατηρήσεών μας είναι

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_n \text{ όπου κάθε } \boldsymbol{x}_\iota \text{ ανήκει στο } R^6$$

Βάση αυτών των παρατηρήσεων θέλουμε να αποκτήσουμε ένα μοντέλο που βασισμένο στις προηγούμενες χρονικά παρατηρήσεις θα μας δίνει προβλέψεις για τη μελλοντική κίνηση των μετοχών.

Για να χειριστούμε τα δεδομένα μας (που είναι σε μορφή χρονοσειράς) στην R κατεβάζουμε τη βιβλιοθήκη xts και ακολούθως χρησιμοποιούμε την εντολή

>GSPC<-as.xts(read.zoo("C:/gspc.csv",sep=",",tz="",header=T, format='%d-%m-%y'))

Για να αποθηκευτούν στην R. Είναι καλό πριν ξεκινήσουμε την ανάλυση του μοντέλου μας να ρίξουμε μια γενική ματιά στα δεδομένα μας.

Καταρχάς ορίζουμε τα δεδομένα μας ως χρονοσειρά που αρχίζει στις 2/1/1970 και τελειώνει στις 15/9/2009. Επίσης παίρνω μόνο την τιμή του Adjusted Close για ανάλυση

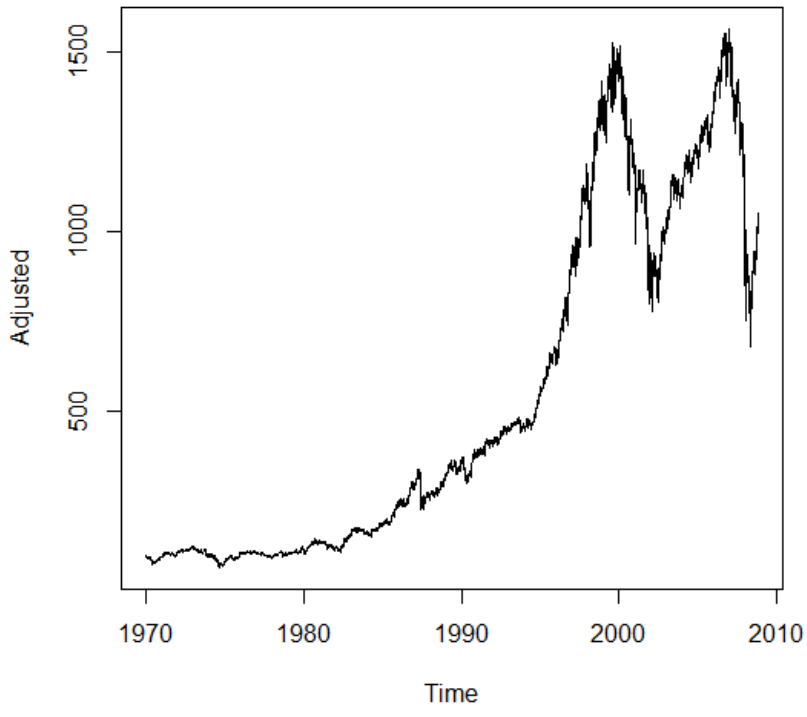
>gspc3<-ts(data=GSPC$Adjusted,start=c(1970,1,2),end=c(2009,9,15)

Στη συνέχεια κάνουμε το γράφημα της προσαρμοσμένης τιμής κλεισίματος και με την εντολή decompose βλέπουμε τις εποχιακές και κυκλικές τάσεις της μετοχής μας.

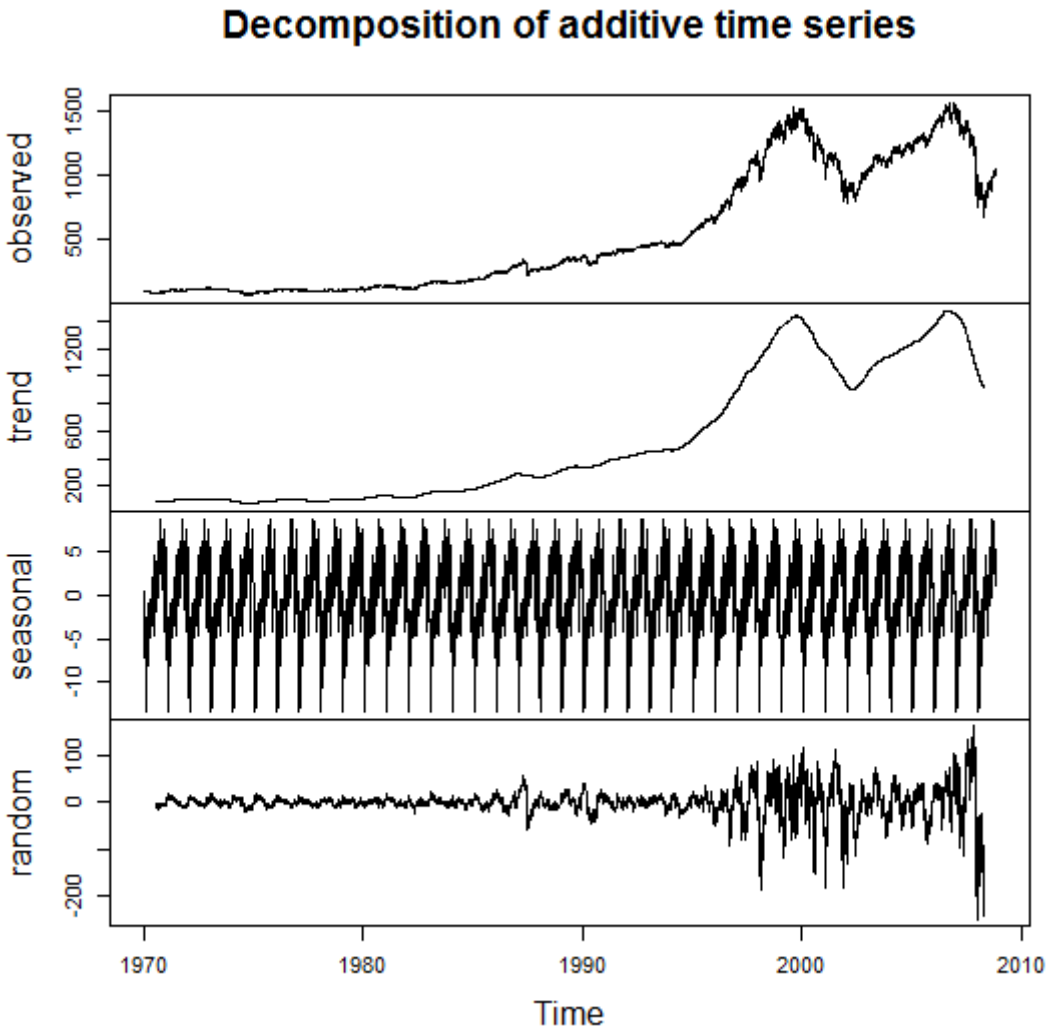
plot(gspc3)

f<-decompose(gspc3)

plot(f)



<u>Γράφημα 2</u>: Κίνηση της μετοχής από το 1970-2009



<u>Γράφημα 3:</u> κίνηση μετοχής, τάση κίνησης μετοχής, εποχιακή τάση και κυκλική τάση μετοχής αντίστοιχα

4.1.2 Δείκτρια συνάρτηση

Σε αυτό το υποκεφάλαιο θα μελετήσουμε ποια από τα δεδομένα μας θα χρησιμοποιήσουμε και σε τι βάθος χρόνου θα επιχειρήσουμε να κάνουμε την πρόβλεψή μας.

Ξεκινώντας κρίνεται αναγκαίο να αναφέρουμε πως αφού ο κύριος ρόλος του data mining στα χρηματοοικονομικά είναι η δημιουργία ενός κερδοφόρου χαρτοφυλακίου στην πράξη δεν είναι τόσο σημαντικό να ξέρουμε απλώς αν την επόμενη μέρα θα παρουσιάσει απλώς αύξηση ή μείωση της μετοχής μας, παρά αν στην διάρκεια των επόμενων κ μερών θα έχουμε αυξητική **τάση** ή πτωτική τάση της μετοχής. Άρα πρέπει να δημιουργήσουμε μια

συνάρτηση που θα μας δείχνει αν κατά τη διάρκεια αυτών των κ ημερών θα έχουμε αυξητική ή πτωτική τάση και θα μας λέει το μέτρο αυτής της ανόδου/καθόδου. Άρα αν αποκτήσουμε αυτό το μέτρο μπορούμε τελικά να καταλήξουμε σε ένα binary πρόβλημα όπου αν η συνάρτηση απόκρισης (από εδώ και πέρα T) είναι αρνητική έχουμε y=0 (sell), ενώ αν είναι θετική y=1(buy) . (Το πλεονέκτημα της T είναι πως πρακτικά δεν κάνουμε καθημερινά αγοραπωλησίες της μετοχής μας λόγω του κόστους φορολόγησης και χρηματιστή. Αντίθετα τις περισσότερες μέρες κάνουμε hold την μετοχή έως ότου παρουσιαστεί κάποια πρόβλεψη για μεγάλη μελλοντική αύξηση ή μείωση του T (|T|>α) και άρα θα έχουμε μεγάλο περιθώριο κέρδους. Στην εφαρμογή αυτή όμως κρίνεται αναγκαίο το πρόβλημα να είναι binary για αυτό και δεν θα ασχοληθούμε στην συνέχεια με την επιλογή hold).

Έστω τώρα η προσέγγιση της μέσης ημερήσιας τιμής

$\bar{P}_i = \frac{C_i+H_i+L_i}{3}$ όπου C,H,L η τιμές Close, High, Low

Και Vi οι k ποσοστιαίες μεταβολές από τη σημερινή μέχρι την k ημέρα.

$$V_i = \{\frac{P_{i+j} - C_i}{C_i}\}_{j=1}^{K}$$

Η δείκτρια συνάρτηση που αναφέραμε πριν θα είναι η

$$T_i = \sum_{v} \{v \in V_i : v > p\% \; U \; v < p\%\}$$

Άρα η Ti με αυτή τη μορφή μας δείχνει περιόδους k ημερών, μεταξύ των οποίων πολλές μέρες έχουν ημερήσια μεταβολή πάνω από το ποσοστό p που θέσαμε εμείς ως σημαντικό για να συναλλαχτούμε με την μετοχή. Προφανώς στην περίπτωσή μας που θέλουμε να καταλήξουμε σε binary μεταβλητή αρκεί να θέσουμε p=0. Γράφουμε αυτή τη συνάρτηση στην R με τη βοήθεια της βιβλιοθήκης DMwR:

```
>T.ind<-function(quotes,tgt.margin=0,n.days=10){
+ v<-apply(HLC(quotes), 1 ,mean)
+ r<-matrix(NA, ncol=n.days,nrow=NROW(quotes))
+ for(x in 1:n.days) r[, x] <-Next(Delt(v, k = x),x)
```

+ x<-apply(r,1,function(x) sum(x[x>tgt.margin | x< -tgt.margin]))

+ if(is.xts(quotes))

+ xts(x, time(quotes))

+ else x}

Για να πάρουμε μια καλύτερη ιδέα για τη συμπεριφορά της δείκτριας συνάρτησης μας κάνουμε τη γραφική των 3 τελευταίων μηνών με τον κώδικα:

>candleChart(last(GSPC,"3 months"), theme="white", TA=NULL)

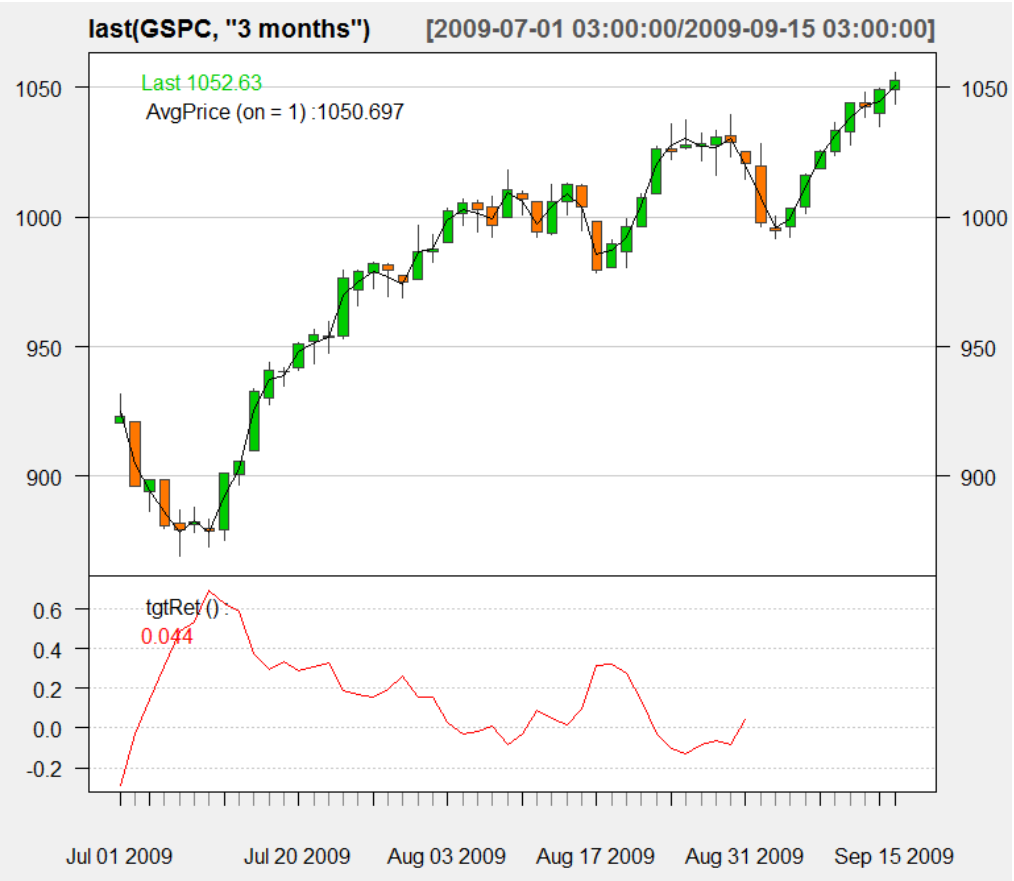
>avgPrice<-function(p) apply(HLC(p), 1, mean)

>addAvgPrice<-newTA(FUN=avgPrice, col=1, legend="AvgPrice")

>addT.ind<-newTA(FUN=T.ind, col="red", legend="tgtRet")

>addAvgPrice(on=1)

>addT.ind()



<u>Γράφημα 4</u>: S&P500 για τους τελευταίους 3 μήνες με την δίκτρια συνάρτηση μας

Η συνάρτηση candleChart() σχεδιάζει την γραφική της μετοχής μας. Η γραφική μας δείχνει ημερήσιες αυξήσεις/ μειώσεις με πορτοκαλί/πράσινο χρώμα αντίστοιχα, όσο μεγαλύτερη η κάθετη μπάρα τόσο μεγαλύτερη η απόλυτη τιμή της Τ. Στην 1η γραφική προσθέσαμε επίσης την μέση τιμή κάθε μέρας. Στην 2η γραφική βλέπουμε την τιμή της συνάρτησης Τ κατά τη διάρκεια αυτών των 3 μηνών. Τέλος είναι εύκολο να παρατηρήσουμε από τη γραφική μας πως οι μεγαλύτερες τιμές της Τ επιτυγχάνονται όταν έχουμε για συνεχόμενες μέρες αύξηση της τιμής της μετοχής.

4.1.3 Ποιες μεταβλητές θα χρησιμοποιήσουμε

Έχοντας υπολογίσει τη δείκτρια συνάρτηση πρέπει να επιλέξουμε βάση ποιων μεταβλητών θα επιχειρήσουμε να κάνουμε την πρόβλεψη μας. Η κύρια υπόθεση πίσω από την πρόβλεψη μας είναι πως η μελλοντική συμπεριφορά των οικονομικών αγορών μπορεί να προβλεφθεί με βάση την συμπεριφορά των προηγούμενων ημερών. Δηλαδή αν υποθέσουμε ότι στο παρελθόν σε αρκετές περιπτώσεις κάποια συμπεριφορά p ακολουθώταν από κάποια συμπεριφορά f τότε στο μέλλον όποτε παρακολουθήσουμε τη συμπεριφορά p θα αναμένουμε να δούμε και την συμπεριφορά f στη συνέχεια. Η δυσκολία σε αυτό το στάδιο είναι να εξάγουμε αρκετές ανεξάρτητες μεταβλητές από τα δεδομένα μας που θα περιέχουν όσο περισσότερες ιδιότητες και πληροφορίες για την δυναμική της μετοχής μας.

Στην R χάρη στο πακέτο TTR μπορούμε να βρούμε πολλές συναρτήσεις που μας δίνουν πληροφορίες για το μοντέλο μας βάση των δεδομένων που θα έχουμε. Για την επιλογή των "σημαντικότερων" συναρτήσεων από αυτές θα ορίσουμε ένα αρχικό σετ συναρτήσεων (καινούργιων δεδομένων που προέρχονται από την ανάλυση των υπαρχουσών) και χρησιμοποιώντας την τεχνική του ***randomForest*** θα εκτιμήσουμε τη σημαντικότητα κάθε καινούργιας παραμέτρου αποκλείοντας τις λιγότερο σημαντικές και κρατώντας τις σημαντικότερες.

Θα επικεντρώσουμε την ανάλυση μας στη τιμή κλεισίματος αφού οι αποφάσεις μας θα παρθούν μετά το κλείσιμο του χρηματιστηρίου. Οι ποσοστιαία μεταβολή μετά από h μέρες υπολογίζεται από

$$R_{i-h} = \frac{C_i - C_{i-h}}{C_{i-h}}$$

Οι μεταβλητές που επιλέξαμε από το πακέτο TTR είναι οι ακόλουθες:

- **ATR: Μέσο πραγματικό εύρος** που είναι ένας δείκτης μεταβλητότητας της σειράς
- **SMI: Στοχαστικός δείκτης ορμής**
- **ADX: Welles Wilder's δείκτης κίνησης**
- **Aroon: Δίκτης Aroon** που προσπαθεί να αναγνωρίσει αρχικές τάσεις
- **BB: Bollinger Bands** που συγκρίνουν τη μεταβλητότητα για μια χρονική περίοδο
- **Chaikin Volatility**
- **CLV: Τιμή κλειστής τοποθεσίας** που συσχετίζει την τιμή κλεισίματος με το ανταλλακτικό του εύρος
- **EMV**: Arm's Ease τιμή μετακίνησης
- **MACD:** ταλαντωτής
- **MFI:** Δίκτης ροής χρήματος
- **SAR**
- **Volatility indicator:** Συντελεστής μεταβλητότητας**.**
- **EMA**
- **RSI**
- **runMean**
- **runSD**
- **Delt**
- **CMO**

Για περαιτέρω πληροφορίες για τις παραπάνω συναρτήσεις παραπέμπουμε στην σελίδα help του πακέτου TTR [17].

```
>myATR<- function(x) ATR(HLC(x))[,"atr"]
>mySMI<- function(x) SMI(HLC(x))[,"SMI"]
>myADX<-function(x) ADX(HLC(x))[,"ADX"]
>myAroon<-function(x) aroon(x[,c("High","Low")])$oscillator
>myBB<-function(x) BBands(HLC(x))[,"pctB"]
>myChaikinVol<-function(x) Delt(chaikinVolatility(x[,c("High","Low")]))[,1]
>myCLV<-function(x) EMA(CLV(HLC(x)))[,1]
```

>myEMV<-function(x) EMV(x[,c("High", "Low", "Close")], x[,"Volume"])

>myMACD<-function(x) MACD(Cl(x))[,2]

>myMFI<-function(x) MFI(x[,c("High","Low","Close")], x[, "Volume"])

>mySAR<-function(x) SAR(x[,c("High","Close")])[,1]

>myVolat<-function(x) volatility(OHLC(x), calc="garman") [,1]

Οι 22 μεταβλητές που περιγράψαμε σχηματίζουν το αρχικό σετ παραμέτρων για την πρόβλεψη της μελλοντικής τιμής του T. Τώρα με τη μέθοδο των Random Forests θα εκτιμήσουμε τη σημαντικότητα της κάθε μεταβλητής που συμμετέχει στη διαδικασία πρόβλεψης. Η σημαντικότητα μπορεί να προβλεφθεί υπολογίζοντας το ποσοστό αύξησης του σφάλματος του Random Forest μετά από την αφαίρεση κάθε μεταβλητής.

Στην προσέγγισή μας στην εφαρμογή θα χωρίσουμε τα δεδομένα μας σε 2 σετ. Το $1^{ο}$ θα είναι το σετ εκπαίδευσης που θα αποτελείται από τις παρατηρήσεις των πρώτων 30 ετών και με το οποίο θα εκπαιδεύσουμε το μοντέλο μας, και το $2^{ο}$ θα είναι το σετ δοκιμής (τελευταία 9 χρόνια), με τα οποία θα ελέγξουμε το ποσοστό επιτυχίας του μοντέλου μας.

Ξεκινώντας κτίζουμε ένα random Forest με τα δεδομένα του σετ εκπαίδευσης:

> library(randomForest)

> data.model<-specifyModel(T.ind(GSPC)~Delt(Cl(GSPC),k=1:10)+

+ myATR(GSPC)+mySMI(GSPC)+myADX(GSPC)+myAroon(GSPC)+CMO(Cl(GSPC))+

+ EMA(Delt(Cl(GSPC)))+myEMV(GSPC)+myVolat(GSPC) +myMACD(GSPC)+

+ myMFI(GSPC)+RSI(Cl(GSPC))+mySAR(GSPC)+runMean(Cl(GSPC))+

+ runSD(Cl(GSPC)))

> rf<-buildModel(data.model,method='randomForest',

+ training.per=c(start(GSPC), index(GSPC["1999-12-31"])),

+ ntree=50,importance=T)

Αξίζει να σημειωθεί πως με το να βάλουμε importance = T το Random Forest υπολόγισε την σημαντικότητα με 2 εναλλακτικές βαθμολογίες. Η $1^{η}$ είναι το ποσοστό του σφάλματος στο δάσος αν αφαιρέσουμε κάθε μεταβλητή. Αυτό μετριέται υπολογίζοντας την αύξηση του μέσου τετραγωνικού σφάλματος μετά από την αφαίρεση της υπό εξέταση μεταβλητής. Αυτή η αύξηση υπολογίζεται κατά μέσο όρο σε όλα τα δέντρα και στη συνέχεια

κοινωνικοποιείται με το se. Η 2η βαθμολογία έχει να κάνει με την μείωση της μη καθαρότητας του κόμβου και πάλι υπολογίζεται κατά μέσο όρο. Στη συνέχεια θα χρησιμοποιήσουμε την 1η βαθμολογία. Αφού αποκτήσαμε το μοντέλο μπορούμε να ελέγξουμε την σημαντικότητα των μεταβλητών με τις παρακάτω εντολές.

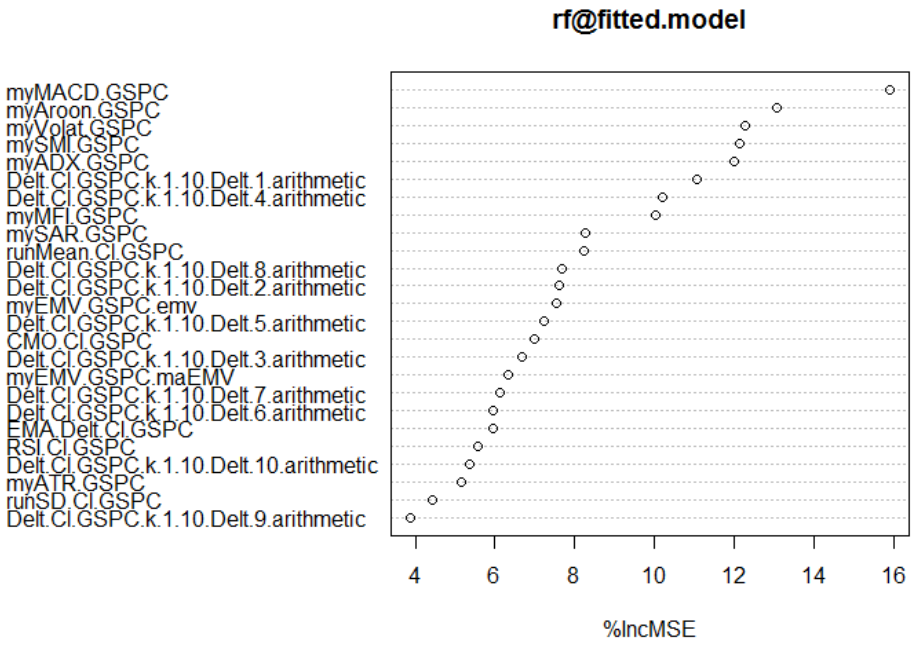
>varImpPlot(rf@fitted.model, type=1)

>imp<-importance(rf@fitted.model, type=1)

>thres<-10

>rownames(imp)[which(imp>thres)]

με την 1η εντολή βλέπουμε την γραφική αναπαράσταση της σημαντικότητας των μεταβλητών μου. Στη συνέχεια θέτουμε ένα αριθμό ως κατώφλι για να επιλέξουμε μόνο κάποιες από τις μεταβλητές μας για να συμμετάσχουν στο μοντέλο μας. Τα ποτελέσματα που παίρνουμε είναι τα εξής.



<u>Γράφημα 5</u>: Σημαντικότητα μεταβλητών

[1] "Delt.Cl.GSPC.k.1.10.Delt.1.arithmetic"

[2] "Delt.Cl.GSPC.k.1.10.Delt.4.arithmetic"

[3] "mySMI.GSPC"

[4] "myADX.GSPC"

[5] "myAroon.GSPC"

[6] "myVolat.GSPC"

[7] "myMACD.GSPC"

[8] "myMFI.GSPC"

Άρα από τις 22 μεταβλητές καταλήξαμε στις 8 πιο σημαντικές. Τελικά ορίζουμε το μοντέλο μας να εξαρτάται από αυτές τις μεταβλητές:

```
> data.model<-specifyModel(T.ind(GSPC)~Delt(Cl(GSPC),k=c(1,4))+
+ +mySMI(GSPC)+myADX(GSPC)+myVolat(GSPC)+
+ + myMACD(GSPC)+myAroon(GSPC)+myMFI(GSPC)).
```

4.2 Λογιστική Παλινδρόμηση

Τώρα που αποκτήσαμε τις μεταβλητές μας μπορούμε να συνεχίσουμε την ανάλυσή μας με την λογιστική παλινδρόμηση.

Όπως είδαμε η μεταβλητή απόκρισής μας T που την παίρνουμε από την συνάρτηση του R T.ind δεν είναι δίτημη (binary). Υπάρχουν πολλοί τρόποι να την κάνουμε δίτημη, κρίνουμε ότι ο ευκολότερος είναι με την βοήθεια της excel, για αυτό και εξάγουμε τις τιμές της T στην excel με την συνάρτηση

```
write.csv(T.ind(GSPC),file="Tind.csv")
```

και στη συνέχεια ανοίγοντας την excel στη διπλανή στήλη θέτουμε τη συνάρτηση (if Bi<0 Ci=0 else Ci=1).Άρα όποτε θα παίρνουμε μεταβλητή 1 θα περιμένουμε αυξητική τάση τις επόμενες μέρες ενώ αν παίρνουμε τιμή 0 πτωτική τάση. Τώρα πολύ απλά με τον τρόπο που δείξαμε στην αρχή του κεφαλαίου επανεισάγουμε την τιμή στην R ως y1.

Έχοντας αποκτήσει και binary μεταβλητή απόκρισης έχουμε ξεπεράσει όλα τα προβλήματα και πάμε να εκπαιδεύσουμε το μοντέλο μας.

```
train<-1:7582
test<-7583:1022
mylogit<-glm(y1~Delt(Cl(GSPC),k=c(1,4))+
+mySMI(GSPC)+myADX(GSPC)+myVolat(GSPC)+
+ myMACD(GSPC)+myAroon(GSPC)+myMFI(GSPC),family="binomial"(link="logit"),
```

subset=train)

thresh<-0.5

predl<-predict(mylogit,newdata=GSPC[test,],type="response")

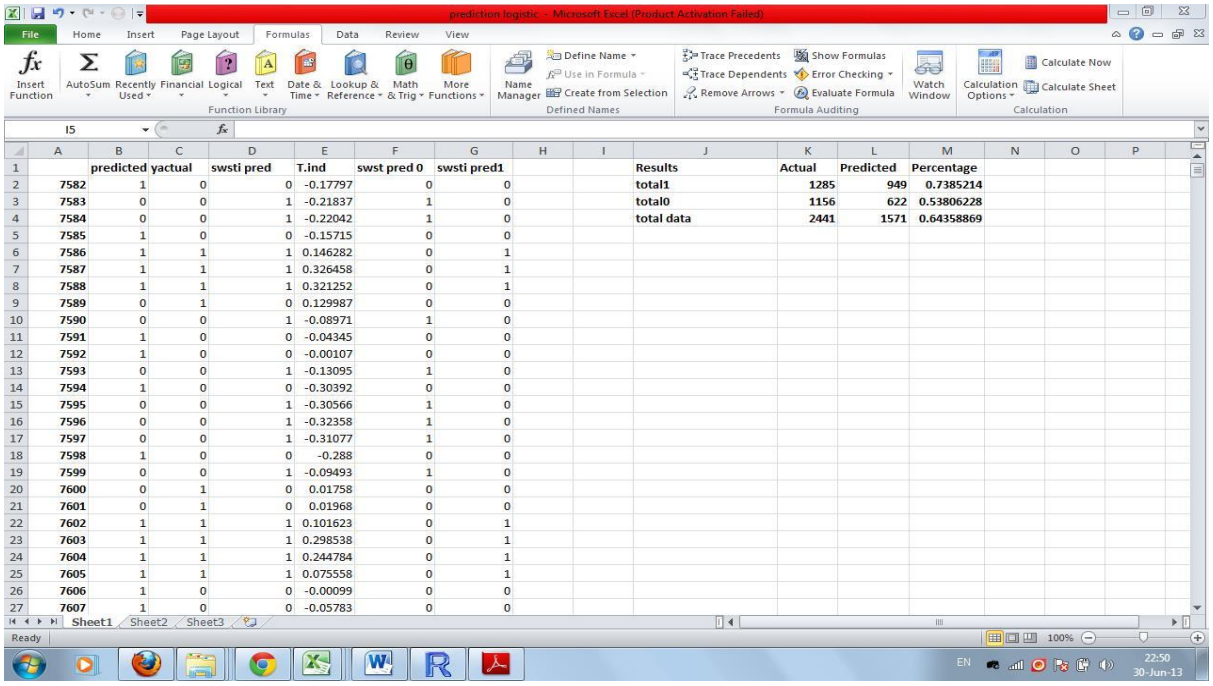
predictionlog<-cut(predl,breaks=c(-Inf,thresh,Inf),labels=c("0","1"))

Σε αυτό το στάδιο, ορίσαμε καταρχάς το σετ εκπαίδευσης και το σετ δοκιμής. (train τα πρώτα 30 χρόνια (7582 δεδομένα) και test τα τελευταία 9). Στη συνέχεια εκπαιδεύσαμε την λογιστική μας παλινδρόμηση στο training subset και μετά κάναμε πρόβλεψη στο test set θέτοντας οποιαδήποτε τιμή κάτω από 0.5 ως 0 και οποιαδήποτε τιμή πάνω από 0.5 ως 1.

Υπάρχουν πολλοί τρόποι για να ελέγξουμε την ακρίβεια των αποτελεσμάτων μας, ο πιο απλός είναι να εξάγουμε όπως και πριν τα δεδομένα μας στην excel και να τα βάλουμε στην στήλη A ως predicted. Δίπλα θα έχουμε στη στήλη B την y1 binary μεταβλητή απόκρισης ως Actual και στην στήλη C θα δημιουργήσουμε συνάρτηση:

Ci={if Bi-Ai=0 then Ci ==1, else Ci}

Με την ίδια ευκολία αφού έχουμε όλα μας τα δεδομένα στην excel μπορούμε να δούμε τον συνολικό αριθμό που το μοντέλο μας προέβλεψε σωστά άνοδο και κάθοδο ανάλογα. Τελικά παίρνουμε το spreadsheet:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		predicted	yactual	swsti pred	T.ind	swst pred 0	swsti pred1			Results	Actual	Predicted	Percentage			
2	7582	1	0	0	-0.17797	0	0			total1	1285	949	0.7385214			
3	7583	0	0	1	-0.21837	1	0			total0	1156	622	0.53806228			
4	7584	0	0	1	-0.22042	1	0			total data	2441	1571	0.64358869			
5	7585	1	0	0	-0.15715	0	0									
6	7586	1	1	1	0.146282	0	1									
7	7587	1	1	1	0.326458	0	1									
8	7588	1	1	1	0.321252	0	1									
9	7589	0	1	0	0.129987	0	0									
10	7590	0	0	1	-0.08971	1	0									
11	7591	1	0	0	-0.04345	0	0									
12	7592	1	0	0	-0.00107	0	0									
13	7593	0	0	1	-0.13095	1	0									
14	7594	1	0	0	-0.30392	0	0									
15	7595	0	0	1	-0.30566	1	0									
16	7596	0	0	1	-0.32358	1	0									
17	7597	0	0	1	-0.31077	1	0									
18	7598	1	0	0	-0.288	0	0									
19	7599	0	0	1	-0.09493	1	0									
20	7600	0	1	0	0.01758	0	0									
21	7601	0	1	0	0.01968	0	0									
22	7602	1	1	1	0.101623	0	1									
23	7603	1	1	1	0.298538	0	1									
24	7604	1	1	1	0.244784	0	1									
25	7605	1	1	1	0.075558	0	1									
26	7606	1	0	0	-0.00099	0	0									
27	7607	1	0	0	-0.05783	0	0									

<u>Spreadsheet 1</u>: Μορφή των δεδομένων μου και αποτελέσματα λογιστικής παλινδρόμησης

Άρα όπως βλέπουμε έχουμε ένα πολύ καλό ποσοστό επιτυχίας που από την κατά 50% σωστή πρόβλεψη που θα κάναμε αν μαντεύαμε τυχαία για την μελλοντική κίνηση των μετοχών μας ανεβάζει στο **64%.** Επίσης προβλέπει σωστά το ότι η μετοχή μου θα αυξηθεί κατά 73%!!!

Results	Actual	Predicted	Percentage
total1	**1285**	**949**	**0.7385214**
total0	**1156**	**622**	**0.53806228**
total data	**2441**	**1571**	**0.64358869**

Πίνακας 3: Αποτελέσματα πρόβλεψης Λογιστικής Παλινδρόμησης

4.3 Νευρωνικά δίκτυα

Χρησιμοποιώντας τις ίδιες εξαρτημένες μεταβλητές έτσι ώστε να είναι δυνατή η εξαγωγή συγκριτικών αποτελεσμάτων μεταξύ των μεθόδων μας θα κάνουμε μοντελοποίηση χρησιμοποιώντας νευρωνικά δίχτυα και συγκεκριμένα το πακέτο της r ***nnet***.

Για να βάλουμε τα δεδομένα μας στην συνάρτηση nnet θα πρέπει να έιναι ορισμένα ως data frame, άρα ως πρώτο βήμα μετατρέπουμε όλες μας τις μεταβλητές σε data frame:

bin.data<-data.frame(Delt(Cl(GSPC),k=1) ,Delt(Cl(GSPC),k=4),mySMI(GSPC),myADX(GSPC) ,myVolat(GSPC),myMACD(GSPC),myAroon(GSPC), myMFI(GSPC),y1)

Έχωντας μετατρέψει τα δεδομένα μας σε data frame χρησιμοποιούμε την συνάρτηση nnet του πακέτου μα για να δημιουργήσουμε το neural network μας:

nn3<-nnet(y1~Delt(Cl(GSPC),k=1) +Delt(Cl(GSPC),k=4)+mySMI(GSPC)+myADX(GSPC) +myVolat(GSPC)+myMACD(GSPC)+myAroon(GSPC)+ myMFI(GSPC), bin.data,subset=train,act.fct="logistic", size=10,maxit=1000000)

Σε αυτή τη συνάρτηση καταρχάς προσδιορίζουμε τη formula, τα δεδομένα δηλαδή που θα χρησιμοποιήσουμε για να εκπαιδεύσουμε το νευρωνικό μας δίκτυο, στη συνέχεια με τον bin.data ορίζουμε από πού θα αντλήσουμε τα δεδομένα μας και με το subset=train ορίζουμε πως θα χρησιμοποιήσουμε μόνο τα πρώτα 30 χρόνια από τα δεδομένα μας. Στη συνέχεια ορίζω ως συνάρτηση ενεργοποίησης τη λογιστική αφού έχουμε δυαδικά δεδομένα.

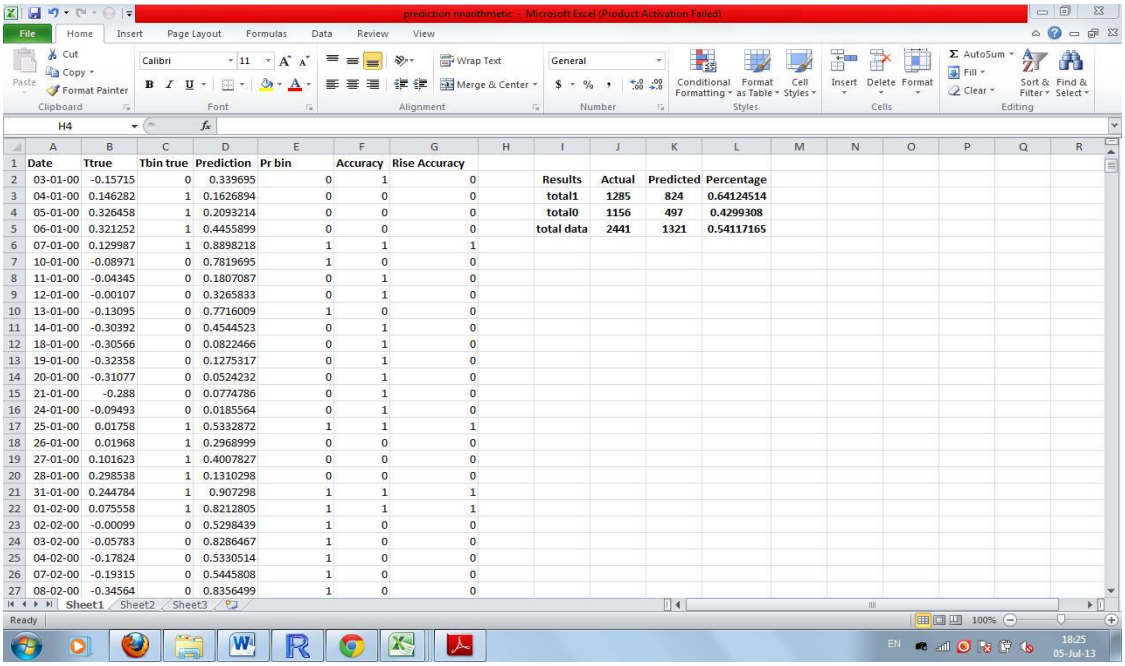
Αφού έχουμε αποκτήσει το νευρωνικό μας δίκτυο ορίζουμε ξανά το σετ δοκιμής λόγω του ότι η prediction στα nnet είναι ευαίσθητη στις μη υπάρχουσες τιμές (NA).

Tdata.eval<-na.omit(as.data.frame(bin.data[test,]))

Και στη συνέχεια κάνουμε την πρόβλεψή μας για τα επόμενα 9 χρόνια

preds<-predict(nn1, Tdata.eval)

ακολουθώντας τώρα ακριβώς την ίδια διαδικασία με τη λογιστική παλινδρόμηση καταλήγουμε στον έξής πίνακα:



	A	B	C	D	E	F	G	H	I	J	K	L
1	Date	Ttrue	Tbin true	Prediction	Pr bin	Accuracy	Rise Accuracy					
2	03-01-00	-0.15715	0	0.339695	0	1	0		Results	Actual	Predicted	Percentage
3	04-01-00	0.146282	1	0.1626894	0	0	0		total1	1285	824	0.64124514
4	05-01-00	0.326458	1	0.2093214	0	0	0		total0	1156	497	0.4299308
5	06-01-00	0.321252	1	0.4455899	0	0	0		total data	2441	1321	0.54117165
6	07-01-00	0.129987	1	0.8898218	1	1	1					
7	10-01-00	-0.08971	0	0.7819695	1	0	0					
8	11-01-00	-0.04345	0	0.1807087	0	1	0					
9	12-01-00	-0.00107	0	0.3265833	0	1	0					
10	13-01-00	-0.13095	0	0.7716009	1	0	0					
11	14-01-00	-0.30392	0	0.4544523	0	1	0					
12	18-01-00	-0.30566	0	0.0822466	0	1	0					
13	19-01-00	-0.32358	0	0.1275317	0	1	0					
14	20-01-00	-0.31077	0	0.0524232	0	1	0					
15	21-01-00	-0.288	0	0.0774786	0	1	0					
16	24-01-00	-0.09493	0	0.0185564	0	1	0					
17	25-01-00	0.01758	1	0.5332872	1	1	1					
18	26-01-00	0.01968	1	0.2968999	0	0	0					
19	27-01-00	0.101623	1	0.4007827	0	0	0					
20	28-01-00	0.298538	1	0.1310298	0	0	0					
21	31-01-00	0.244784	1	0.907298	1	1	1					
22	01-02-00	0.075558	1	0.8212805	1	1	1					
23	02-02-00	-0.00099	0	0.5298439	1	0	0					
24	03-02-00	-0.05783	0	0.8286467	1	0	0					
25	04-02-00	-0.17824	0	0.5330514	1	0	0					
26	07-02-00	-0.19315	0	0.5445808	1	0	0					
27	08-02-00	-0.34564	0	0.8356499	1	0	0					

<u>Spreadsheet 2</u>: Μορφή των δεδομένων μου και αποτελέσματα νευρωνικών δικτύων

Με πίνακα συνοπτικών αποτελεσμάτων

Results	Actual	Predicted	Percentage
total1	**1285**	**824**	**0.64124514**
total0	**1156**	**497**	**0.4299308**
total data	**2441**	**1321**	**0.54117165**

<u>Πίνακας 4:</u> Αποτελέσματα πρόβλεψης Νευρωνικού Δικτύου

Όπως παρατηρούμε αν και με το νευρωνικό δίκτυο έχουμε ένα αρκετά καλό ποσοστό επιτυχίας όσον αφορά τις περιπτώσεις που ο δείκτης μας παρουσιάζει αύξηση, η πρόβλεψη μας για τις περιπτώσεις που παρουσιάζει μείωση ο δείκτης είναι της τάξης του 42% που είναι πολύ μικρή. Αυτό εξηγείται με το ότι ενώ το πραγματικό πλήθος των ημερών με αυξητική τάση είναι 1285 το νευρωνικό μου δίκτυο προβλέπει συνολικά 1490 ημέρες με αυξητική τάση. Άρα είναι λογικό πως θα έχει περισσότερο ποσοστό επιτυχίας στην εύρεση αυξητικής τάσης. Επίσης αν και η συνολική μας ακρίβεια φαίνεται αρκετά χαμηλή σε σχέση με αυτή της Λογιστικής παλινδρόμησης δεν είναι καθόλου άσχημη για αυτού του είδους την στατιστική ανάλυση αφού όπως αναφέραμε και πριν δεν θα δράσουμε αποκλειστικά με την πληροφορία που μας δίνουν τα μοντέλα μας αλλά θα χρησιμοποιήσουμε την πληροφορία ως δείκτη για την ενέργειά μας.

4.4 Boosting

Σε αυτό το μέρος της εφαρμογής θα μοντελοποιήσουμε τα δεδομένα μας χρησιμοποιώντας το boosting με δέντρα αποφάσεων ως μοντέλα εκμάθησης βάσης. Καταρχάς θα ορίσουμε τις μεταβλητές μας ως V1,V2 …. V8 για να αποφευχθούν προβλήματα στην πρόβλεψη:

```
V1<-Delt(Cl(GSPC),k=1)
V2<-Delt(Cl(GSPC),k=4)
V3<-mySMI(GSPC)
V4<-myADX(GSPC)
V5<-myVolat(GSPC)
V6<-myMACD(GSPC)
V7<-myAroon(GSPC)
V8<-myMFI(GSPC)
bin1.data<-data.frame(V1,V2,V3,V4,V5,V6,V7,V8,y)
```

Στη συνέχεια χρησιμοποιώντας τη βιβλιοθήκης ada της R δημιουργούμε το μοντέλο μας βάση της θεωρίας που αναπτύξαμε στο Boosting με τον εξής κώδικα:

```
greal50<-ada(y~V1+V2+V3+V4+V5+V6+V7+V8,bin1.data,iter=50,nu=0.001,type="real",
 bag.frac=1,model.coef=FALSE,loss="logistic",
 control=rpart.control(maxdepth=2,cp=-1,minsplit=0),subset=train)
```

Όπου χρησιμοποιούμε τον real AdaBoost με η(p)=log$\frac{p}{1-p}$ με το p να ανήκει στο [0,1] και συνάρτηση κόστους τη Λογιστική. Επίσης ορίσαμε ότι ο αλγόριθμος μας θα κάνει 50 επαναλήψεις. Στη συνέχεια αν καλέσουμε το greal παίρνουμε τα εξής αποτελέσματα:

```
> greal50
Call:
ada(y ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8, data = bin1.data,
   iter = 50, nu = 0.001, type = "real", bag.frac = 1, model.coef = FALSE,
   loss = "logistic", control = rpart.control(maxdepth = 2,
      cp = -1, minsplit = 0), subset = train)
Loss: logistic Method: real   Iteration: 50
Final Confusion Matrix for Data:
        Final Prediction
True value    0    1
       0       889 2416
       1        293 3983
Train Error: 0.357
Out-Of-Bag Error:  0  iteration= 6
Additional Estimates of number of iterations:
train.err1 train.kap1
        1         1
```

Που μας δείχνει τις σωστές προβλέψεις που κάνει το boosting στα 7581 δεδομένα του σετ εκπαίδευσης . Προσέχουμε επίσης ότι το out of-bag Error είναι 0 αφού θέσαμε bag.frac=1, δηλαδή δεν κάναμε Stochastic gradient Boosting αλλά απλό boosting.

Έχοντας δει το real Adaboost με παρόμοιο ακριβώς τρόπο κάνουμε και το gentle Adaboost:

```
ggen50<-ada(y~V1+V2+V3+V4+V5+V6+V7+V8,bin1.data,iter=50,
type="gentle",bag.frac=1,model.coef=FALSE,loss="logistic",
 control=rpart.control(maxdepth=2,cp=-1,minsplit=0),subset=train)
```

με αποτελέσματα:

Call:

ada(y ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8, data = bin1.data,

iter = 50, type = "gentle", bag.frac = 1, model.coef = FALSE,

loss = "logistic", control = rpart.control(maxdepth = 2,

cp = -1, minsplit = 0), subset = train)

Loss: logistic Method: gentle Iteration: 50

Final Confusion Matrix for Data:

Final Prediction

True value 0 1

0 1686 1619

1 835 3441

Train Error: 0.324

Out-Of-Bag Error: 0 iteration= 6

Additional Estimates of number of iterations:

train.err1 train.kap1

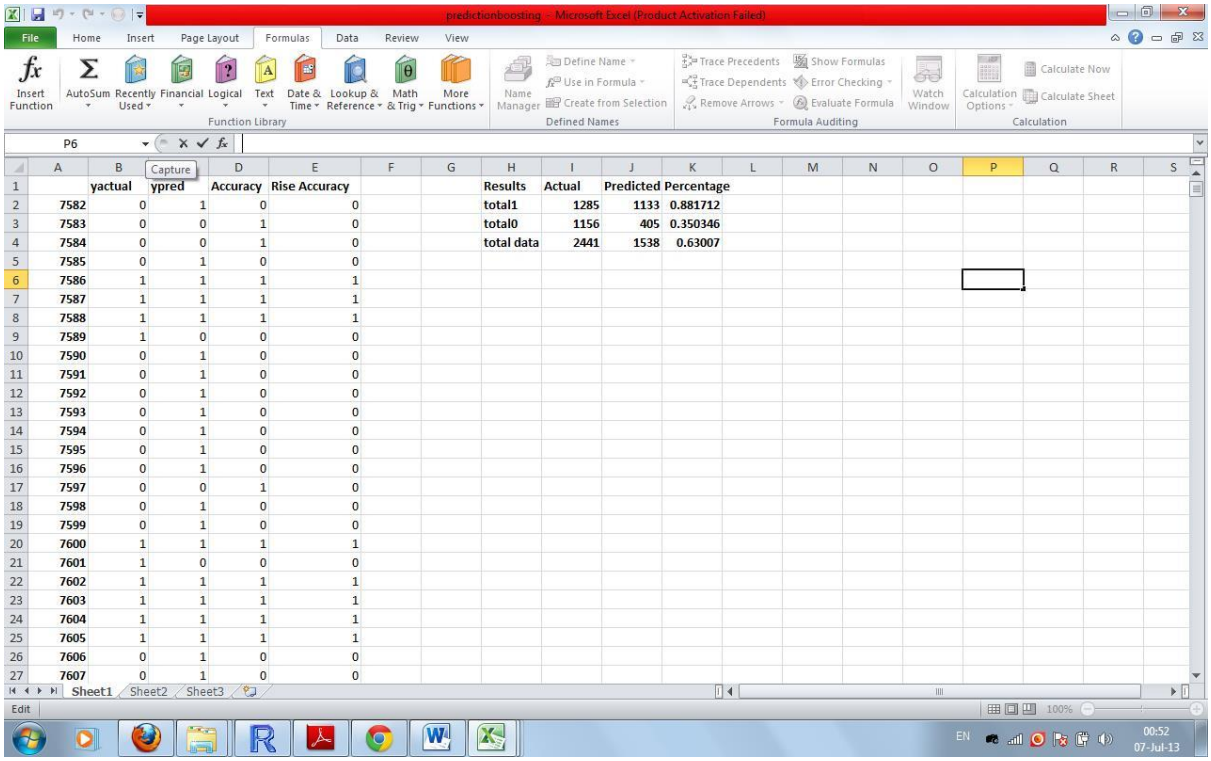
50 1

Τώρα που είδαμε τα 2 μας μοντέλα πάμε να δούμε την ακρίβεια πρόβλεψης στο σετ δοκιμής. Ακολουθούμε ακριβώς την ίδια μέθοδο με τα δύο προηγούμενα μοντέλα και εγγράφουμε την πρόβλεψη μας σε ένα αρχείο excel.

predgen<-predict(ggen50,bin.data[test,])

predreal<-predict(greal50,bin.data[test,])

Για τον Real Adaboost έχουμε



	A	B	C	D	E	F	G	H	I	J	K
1		yactual	ypred	Accuracy	Rise Accuracy			Results	Actual	Predicted	Percentage
2	7582	0	1	0	0			total1	1285	1133	0.881712
3	7583	0	0	1	0			total0	1156	405	0.350346
4	7584	0	0	1	0			total data	2441	1538	0.63007
5	7585	0	1	0	0						
6	7586	1	1	1	1						
7	7587	1	1	1	1						
8	7588	1	1	1	1						
9	7589	1	0	0	0						
10	7590	0	1	0	0						
11	7591	0	1	0	0						
12	7592	0	1	0	0						
13	7593	0	1	0	0						
14	7594	0	1	0	0						
15	7595	0	1	0	0						
16	7596	0	1	0	0						
17	7597	0	0	1	0						
18	7598	0	1	0	0						
19	7599	0	1	0	0						
20	7600	1	1	1	1						
21	7601	1	0	0	0						
22	7602	1	1	1	1						
23	7603	1	1	1	1						
24	7604	1	1	1	1						
25	7605	1	1	1	1						
26	7606	0	1	0	0						
27	7607	0	1	0	0						

<u>Spreadsheet 3</u>: Μορφή των δεδομένων μου και αποτελέσματα real Adaboost

Με πίνακα συνοπτικών αποτελεσμάτων

Results	Actual	Predicted	Percentage
total1	**1285**	**1133**	**0.881712**
total0	**1156**	**405**	**0.350346**
total data	**2441**	**1538**	**0.63007**

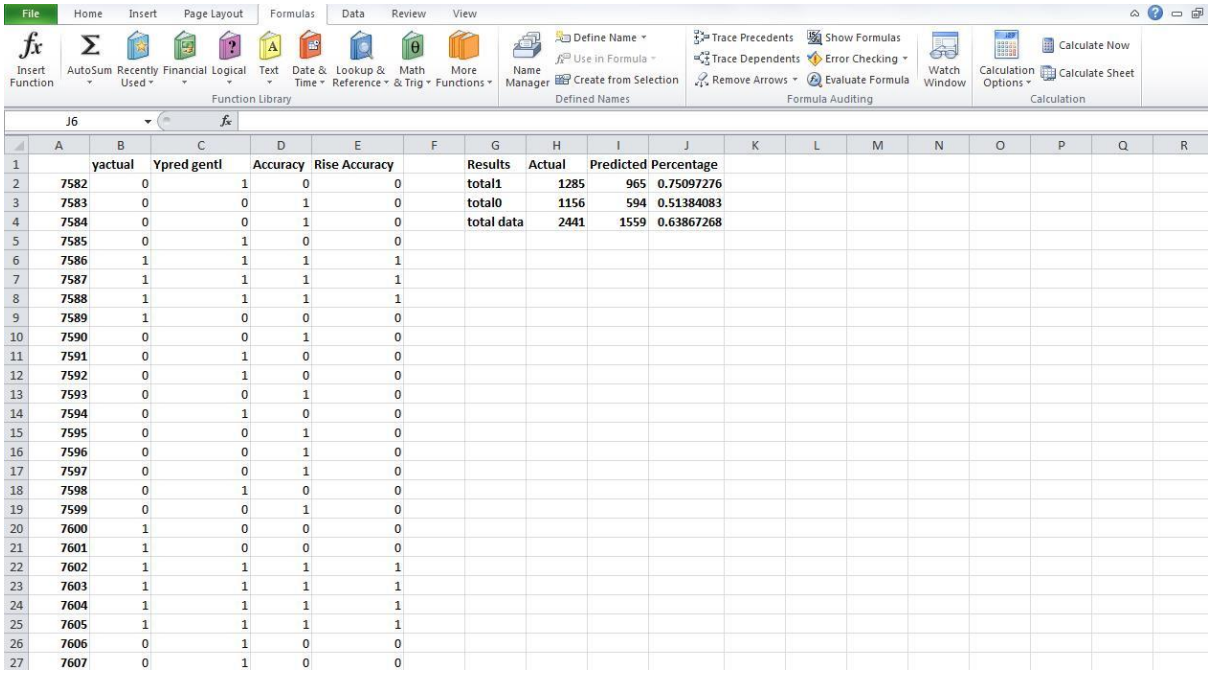
<u>Πίνακας 5:</u> Αποτελέσματα πρόβλεψης real AdaBoost

Όπως βλέπουμε ο Real Adaboost πετυχαίνει μια αρκετά καλή ακρίβεια της τάξης του 63% με 88% σωστή πρόβλεψη ανόδου αλλά μόνο 35% σωστή πρόβλεψη πτώσης. Από την άλλη για τον Gentle Boost έχουμε τον συνοπτικό πίνακα αποτελεσμάτων:

Results	Actual	Predicted	Percentage
total1	**1285**	**965**	**0.75097276**
total0	**1156**	**594**	**0.51384083**
total data	**2441**	**1559**	**0.63867268**

<u>Πίνακας 6:</u> Αποτελέσματα πρόβλεψης gentle AdaBoost

Με ελαφρά καλύτερη ακρίβεια πρόβλεψης αλλά σαφώς πιο ισοζυγισμένη πρόβλεψη ανόδου/καθόδου ακόμα και αν πάλι υπερτερεί η σωστή πρόβλεψη ανοδικής τάσης.



	A	B	C	D	E	F	G	H	I	J
1		yactual	Ypred gentl	Accuracy	Rise Accuracy		Results	Actual	Predicted	Percentage
2	7582	0	1	0	0		total1	1285	965	0.75097276
3	7583	0	0	1	0		total0	1156	594	0.51384083
4	7584	0	0	1	0		total data	2441	1559	0.63867268
5	7585	0	1	0	0					
6	7586	1	1	1	1					
7	7587	1	1	1	1					
8	7588	1	1	1	1					
9	7589	1	0	0	0					
10	7590	0	0	1	0					
11	7591	0	1	0	0					
12	7592	0	1	0	0					
13	7593	0	0	1	0					
14	7594	0	1	0	0					
15	7595	0	0	1	0					
16	7596	0	0	1	0					
17	7597	0	0	1	0					
18	7598	0	1	0	0					
19	7599	0	0	1	0					
20	7600	1	0	0	0					
21	7601	1	0	0	0					
22	7602	1	1	1	1					
23	7603	1	1	1	1					
24	7604	1	1	1	1					
25	7605	1	1	1	1					
26	7606	0	1	0	0					
27	7607	0	1	0	0					

Spreadsheet 4: Μορφή των δεδομένων μου και αποτελέσματα gentle Adaboost

4.5 Σύνοψη Εφαρμογής και Συμπεράσματα:

Τελικά, ξεκινώντας από τις ημερήσιες τιμές κλεισίματος, ανοίγματος κτλ του δείκτη Standar & Poor 500 βρήκαμε μία δίκτρια συνάρτηση που μας έδειχνε την τάση του δείκτη μας τις επόμενες μέρες. Βάση αυτού του δείκτη δημιουργήσαμε μια binary μεταβλητή και εισάγαμε κάποιες συναρτήσεις, μεταβλητές από το πακέτο DMwR ως επεξηγηματικές μεταβλητές. Χρησιμοποιώντας τη μέθοδο Rain Forest του Bagging διαλέξαμε ποιες από αυτές τις μεταβλητές θα κρατήσουμε στη μοντελοποίηση μας και τις κρατήσαμε για όλα τα μοντέλα ώστε να μπορέσουμε τελικά να συγκρίνουμε τα αποτελέσματα μας. Τα αποτελέσματα μας δείχνουν τεράστιο ενδιαφέρον αφού όλα μας τα μοντέλα (πλην του νευρωνικού διχτύου) πέτυχαν ακρίβεια πρόβλεψης πάνω από 60%. Με καλύτερη ακρίβεια στο συγκεκριμένο πρόβλημα να έχει η Λογιστική παλινδρόμηση με 64% και να ακολουθά το gentle και real boosting με 63,8% και 63,0% αντίστοιχα. Το νευρωνικό μας δίκτυο αν και είχε πολύ πιο χαμηλό ποσοστό πρόβλεψης (54%) δεν μας απογοήτευσε αφού στα προβλήματα πρόβλεψης κίνησης μετοχών ακόμα και ένα ποσοστό επιτυχίας αυτής της τάξης θεωρείται ικανοποιητικό.

Κλείνοντας θέλω να τονίσω το πόσο επιτυχημένη ήταν η χρήση τόσο της μεθόδου Bagging για επιλογή παραμέτρων όσο και του Boosting στην πρόβλεψη της μελλοντικής τάσης του δείκτη μας αφού η ακρίβεια που πετύχαμε ήταν κάτι παραπάνω από ικανοποιητική. Αξίζει να σημειώσουμε επίσης πως ποσοστό αυτής της ακρίβειας οφείλεται στην Real και Gentle επέκταση του Adaboost που μας παρείχαν σαφώς βελτιωμένη ακρίβεια.

Βιβλιογραφία

1. Alfaro E., Gamez M., Garcia N. (2012) *adabag:An R package for classification with Adaboost.M1, AdaBoost-SAMME and Bagging*
2. Aslam, Javed A , Popa , Raluca A. , and Rivest, Ronald L. (2007), *On Estimating the Size and Confidence of a Statistical Audit*, Proceedings of the Electronic Voting Technology Workshop (EVT '07), Boston, MA, August 6, 2007
3. Bishop C.M. (1995), *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.
4. Breiman L. (1996) *Bagging predictors"* Machine Learning 24(2): 123-140. Doi:10.1007/BF00058655
5. Breimann L., Cutler A. , Liaw A. , Wiener M. (2012) *randomForest: Breiman and Cutler's random forests for classification and regression*
6. Breiman L. , Friedman J. H , Olshen R. A. and Stone C. J. , (1984). *Classification and Regression Trees*, Wadsworth, Belmont.
7. Cao D.S. , Xu Q.S., Liang T.Z., Zhang L.X., Li H.D., (2010) *The boosting: A new idea for building Models,* Chemometrics and Intelligent Laboratory Systems 100(2010) 1-11
8. Codreanu D.E, Popa I. , Parpandel D. (2011), *Accounting and Financial Data Analysis, Data Mining Tools ,* 6th edition of the international conference European Integration Realities And Perspectives
9. Collet D. (2003). *Modelling Binary Data,* 2nd Edition, London: Champman and Hall
10. Cowpertwait P.S.P. (2006) *Introductory Time Series with R,* Springer Science and Business Media
11. Culp M., Johnson K., Michailidis G. ,(2006) *ada: an R Package for Stochastic Boosting,* Journal of Statistical Software, volume 17, 2nd issue
12. Dass R. ,(2006) *Data Mining in Banking and Finance: A Note for Bankers,* Indian Institute of Management Ahmedabad
13. Fawcett T.: (2003). *ROC Graphs: Notes and practical considerations for Data Mining researchers,* Intelligent Enterprise Technologies Laboratory
14. Fayyard U. , Shapiro G.P. , Smyth P. (1996), *From Data Mining to Knowledge Discovery Databases*

15. Freund Y. , Schaphire R.E. (1997) *A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting.* Journal of Computer and System Sciences.
16. Freund Y., Schapire R.E. (1999), *A short introduction to Boosting,* Journal of Japanese Society for Artificial Intelligence
17. Gurney K., (1997). *An introduction to Neural Networks*, University of Sheffield.
18. Hastie T. , Tibshirani R. and Friedman J., (2001*), The Elements of Statistical Learning, Data Mining Inference and Prediction*, Springer New York.
19. Hosmer D.W., Lemeshow Jr. S. (2004), *Applied Logistic Regression,* John Wiley and Sons
20. Kamath C., (2009). *Scientific Data Mining, a Practical Perspective*. Philadelphia: Society for Industrial and Applied Mathematics.
21. Kantardzic M. (2003) *Data Mining: Concepts, Models, Methods and Algorithms,* JohnWiley & Sons. ISBN 0-471-22852-4 OCLC 5005536
22. Kovalerchuk B. , Vityaev E.,(2010) *Data Mining for Financial Applications ,* Data Mining and Knowledge Discovery Handbook 1153-1169
23. Kuncheva L.I. (2004) *Combining Pattern Classifiers, Methods and Algorithms*, A John Wiley & Sons, inc, publication
24. Langdell S., NAG,(2002) *Examples of the use of data mining in financial applications,* Financial Engineering News
25. Larose D.T., (2005). *Discovering Knowledge in Data. An Introduction to Data Mining*, John Wiley and Sons, Hoboken, New Jersey.
26. Lee T.H. , Yang Y. ,(2004) *Bagging Binary Predictors for Time Series,* Econometric Society Far Eastern Meeting
27. Lemmens A. , Croux C., (2006) *Bagging and Boosting Classification Trees to predict Churn,* Journal of Marketing Research
28. Long P.M. , Servedio R.A. , (2010) *Random Classification Noise Defeats All Convex Potential Boosters*
29. Ripley B.D. .(1996) *Pattern Recognition and Neural Networks,* University Press, Cambridge
30. Samworth R.J (2012), *Optimal weighted nearest neighbor classifiers"* Annals of Statistics

31. Schaphire E.R. (1990) *The Strength of the weak Learnability,* Machine Learning (Boston, MA: Kluwer Academic Publishers) 5 (2) :197-227
32. Shapiro P. , Gregory, Parker , Gary , (2011) *Lesson: Data Mining and Knowledge Discovery: An Introduction,* Introduction to Data Mining, KD Nuggets
33. Torgo L. , (2010)*Data Mining with R :learning with Case studies,* Chapman & Hall/CRC
34. Ulrich J.(2013) , *TTR: Technical Trading Rules*
35. Venables W.N. and Ripley B.D. (2001) *nnet: Feed-forward Neural Networks and Multinomial Log-Linear Models*
36. Wuertz D. , Chalabi Y. (2001) *time series: Rmetrics – Financial Time Series Object*
37. Zeileis A. ,Grothendiek G., Ryan J.A, Andrews F., *zoo: S3 Infrastructure for Regular and Irregular Time Series (Z's ordered observation)*
38. Zheng Z., (2006) *,Boosting and Bagging of Neural Networks with Applications to Financial Time Series*
39. Zivot E. (2008) , *Working with time series Data in R*
40. Μπακόπουλος Α., Χρυσοβέργης Ι. (2009) *Εισαγωγή στη Αριθμητική Ανάλυση,* Εκδόσεις Συμεων
41. Οικονόμου Π. , Καρωνη Χ. (2010) , *Στατιστικα Μοντέλα Παλινδρομησης,* Εκδοσεις Συμεων Αθηνα 2010
42. Ρίζος Γ. , (1996). *Τεχνητά νευρωνικά δίκτυα*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.