ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

BAGGING, BOOSTING και SVM ΜΕΘΟΔΟΙ ΜΕ ΕΦΑΡΜΟΓΗ ΣΤΑ ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΑ

Μάριος Παός

Επιβλέπων Καθηγητής: Χ.ΚΟΥΚΟΥΒΙΝΟΣ

ΑΘΗΝΑ 2014

Περίληψη

Σε μια εποχή όπου, ο αριθμός των συνόλων δεδομένων που μας περιβάλλει έχει πάρει τεράστιες διαστάσεις, καλούμαστε να αναλύσουμε αυτά τα σύνολα δεδομένων με σκοπό την εξαγωγή χρήσιμών πληροφοριών και την λήψη αποφάσεων. Λόγο του μεγέθους των δεδομένων καθώς και το ότι σύνολα δεδομένων από σύνολα δεδομένων διαφέρουν ως προς το μέγεθος το τύπο των παραμέτρων και την μορφή, καλούμαστε να βρούμε καινούργιες μεθόδους για αυτοματοποιημένη ανάλυση δεδομένων που θα έχουν ικανότητες βελτίωσης και προσαρμοστικότητας. Έτσι στρεφόμαστε στο τομέα της τεχνητής νοημοσύνης για τις κατάλληλες λύσεις και ειδικότερα στις μηχανές εκμάθησης. Ακριβέστερα στην παρούσα διπλωματική εργασία θα επικεντρωθούμε στις μεθόδους Bagging, Boosting και στις μηχανές διανυσματικής υποστήριξης (support vector machine SVM) καθώς επίσης και στο συνδυασμό των μηχανών διανυσματικής υποστήριξης με το Bagging και το Boosting.

Στο πρώτο κεφάλαιο παρουσιάζονται εκτενώς η εξόρυξη δεδομένων και οι μηχανές εκμάθησης, οι εφαρμογές που έχουν, οι υποκατηγορίες στις οποίες χωρίζονται και ο τρόπος λειτουργίας τους.

Στο δεύτερο κεφάλαιο γίνεται εκτενής ανάλυση των μηχανών διανυσματικής υποστήριξης. Ειδικότερα αναλύεται ο ακριβής τρόπος λειτουργίας τους παρουσιάζοντας το μαθηματικό υπόβαθρο πάνω στο οποίο είναι κατασκευασμένες, ενώ γίνεται η απόδειξη κάποιων βασικών σχέσεων. Στην συνέχεια με την χρήση της R γίνεται μια εφαρμογή σε πρόβλημα ταξινόμησης και σχολιασμός των αποτελεσμάτων.

Στο τρίτο κεφάλαιο παρουσιάζονται οι μέθοδοι Boosting και Bagging περιγράφοντας την γενική τους ιδέας, πλεονεκτήματα μειονεκτήματα και στο τι αποσκοπούν. Ακόμη παραθέτουμε και περιγράφουμε τις διάφορες παραλλαγές των αλγορίθμων τους. Στο τέλος και πάλι με την χρήση της R εφαρμόζουμε αυτές τις μεθόδους στο ίδιο σύνολο δεδομένων και κάνουμε μια σύγκριση των δύο μεθόδων.

Στο τέταρτο κεφάλαιο χρησιμοποιούμε Boosting και Bagging έχοντας ως βασικό ταξινομητή μηχανές διανυσματικής υποστήριξης. Γίνονται οι κατάλληλες τροποποιήσεις των αλγορίθμων και βλέπουμε την αποτελεσματικότητα των μεθόδων αυτών σε μια σειρά πειραμάτων εξάγοντας χρήσιμα αποτελέσματα και συμπεράσματα.

Στο πέμπτο κεφάλαιο γίνεται η εφαρμογή των πιο πάνω μεθόδων για την πρόβλεψη της κίνησης του χρηματιστηριακού δείκτη FTSE 100. Γίνεται ανάλυση των δεδομένων μας και με την χρήση καταλλήλων πακέτων στην R αφού εκπαιδεύσουμε τους αλγορίθμου μας προβλέπουμε την κίνηση του δείκτη μας. Τέλος γίνεται και πάλι σύγκριση των αποτελεσμάτων μας.

Στο έκτο και τελευταίο κεφάλαιο γίνεται ένας επίλογος όπου παραθέτουμε τα γενικά μας συμπεράσματα.

Abstract

At a time when the numbers and sizes of datasets that surrounds us has gotten huge dimensions, we have to analyze it in order to extract useful information and decisions. Due to the huge size of datasets and the differences between the type of parameters and the form of these datasets, we are obliged to find new methods for automated analysis of data that will improve skills and adaptability. So we look into the fields of artificial intelligence for appropriate solutions and in particular on machines learning methods. More precisely in this degree thesis we will focus on methods of Bagging, Boosting and Support Vector Machines (SVM) as well as the combinations of support vector machines with Bagging and Boosting.

The first chapter presents an extensive data mining and machine learning applications, and it also presents the subcategories which this methods are divided and modus operandi.

The second chapter is an extensive analysis of support vector machines. As specifically discussed the precise mode, presents the mathematical background and the proof of some basic relations. Then, with the use of R, there is an application to a classification problem and a discussion of the results.

The third chapter presents the methods of Boosting and Bagging describing their general concept, their advantages and disadvantages and the aim of these methods. The chapter also lists and describes the various variants of algorithms. At the end with the use of R we apply these methods to the same data set and we make a comparison of two methods.

In the fourth chapter we use Boosting and Bagging having as base learner support vector machines. They make suitable modifications of the algorithms and we see the effectiveness of these methods in a series of experiments extracting useful results and conclusions.

The fifth chapter is the application of the above methods for predicting the movement of the stock index FTSE 100. By analyzing our data and with the use of suitable packages in R, we train our algorithms and then predicts the movement of our index. Finally we compare the results again.

The sixth and final chapter is an epilogue and it presents our general conclusions.

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής αυτής εργασίας νιώθω την υποχρέωση να εκφράσω τις ιδιαίτερες ευχαριστίες μου στα άτομα που συνέβαλαν στην εκπόνηση της.

Πρωτίστως θα ήθελα να ευχαριστήσω θερμά τον Καθηγητή του ΕΜΠ κ. Χρήστο Κουκουβίνο για την ανάθεση της εργασίας αυτής και κυρίως για τη χρήσιμη καθοδήγηση και εμπιστοσύνη που μου έδειξε κατά την διάρκεια εκπόνηση της παρούσας εργασίας.

Ιδιαίτερες ευχαριστίες οφείλονται στην υποψήφιο Διδάκτορα Κρυσταλλένια Δρούσου για την συνεχή υποστήριξη, καθοδήγηση, υπομονή και εμπιστοσύνη που μου έδειξε καθ' όλη τη διάρκεια της συνεργασίας μας.

Επίσης θέλω να εκφράσω τις βαθύτατες ευχαριστίες στην οικογένεια μου για την στήριξη τους σε όλη την διάρκεια των σπουδών μου και την αμέριστη συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια. Τέλος θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου.

Contents

Περιεχόμενα σχημάτων.

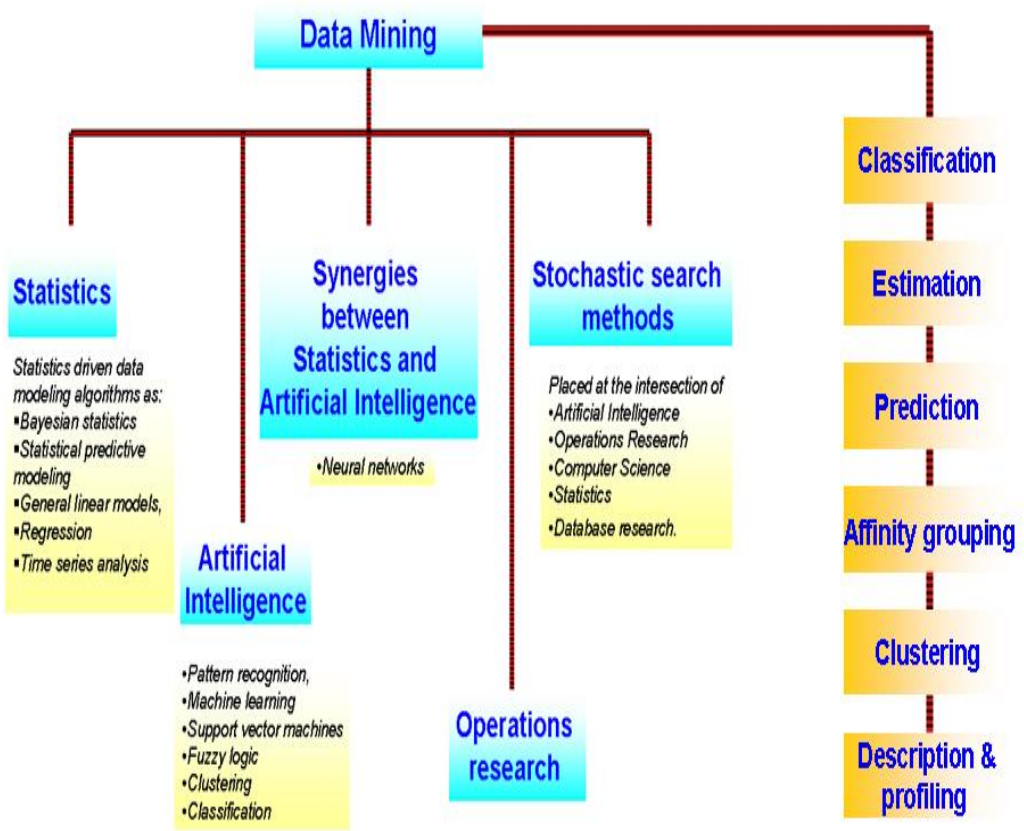
Περιεχόμενα πινάκων.

1. Εξόρυξη δεδομένων (data mining)

1.1 Εισαγωγή

Με την τεράστια ποσότητα δεδομένων που είναι αποθηκευμένα σε αρχεία, βάσεις δεδομένων και σε διάφορα άλλα αποθηκευτικά μέσα γίνεται επιτακτική η ανάγκη για ανάπτυξη ισχυρών μέσων για την ανάλυση και ερμηνεία των δεδομένων αυτών, καθώς και η ανάγκη για εξαγωγή μοτίβων (pattern recognition) και χρήσιμων πληροφορίων, που δεν γνωρίζαμε πριν την ανάλυση αυτή, και θα μας βοηθήσουν στην διαδικασία λήψης αποφάσεων.

Η εξόρυξη δεδομένων (data mining) αναφέρεται ακριβώς σε αυτή την εξόρυξη πληροφοριών που ήταν άγνωστες προηγουμένως και που πιθανόν να φανούν χρήσιμες.



ΣΧΗΜΑ 1-1:DATA MINING

Το data mining περιλαμβάνει μια σειρά από τεχνικές από διάφορους κλάδους όπως: οι μηχανές εκμάθησης (machine learning), η στατιστική, η αναγνώριση μοτίβων (pattern recognition) , τα τεχνητά νευρωνικά δίκτυα (neural network), η χωρική και χρονική ανάλυση δεδομένων κ.α.

Το data mining μπορεί να χρησιμοποιηθεί για προβλήματα:

- Κατασκευής μοντέλων πρόβλεψης (παλινδρόμησης και ταξινόμησης)
- Κατάτμησης (data clustering)
- Περίληψης
- Ανάλυσης ακραίων παρατηρήσεων (outlier analysis)
- Ανάλυσης διασποράς
- Ανάλυσης συσχέτισης

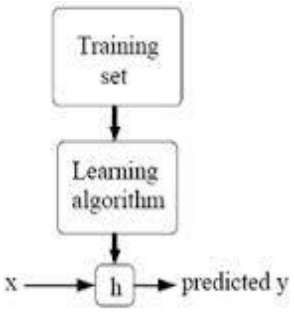
τα παραπάνω περιγράφονται αναλυτικά στα επόμενα κεφάλαια.

Είναι αποδεκτό ότι οι χρήστες δεν έχουν μια σαφή εικόνα για το τι πληροφορίες ή μοτίβα θέλουν να εξορύξουν από τα δεδομένα τους, οπότε είναι σημαντικό να έχουμε ένα ευέλικτο σύστημα data mining το οποίο θα δίνει την δυνατότητα εξόρυξης διαφορετικών ειδών πληροφορίας.

Όπως προείπαμε το data mining μπορεί να εξάγει, από την βάση δεδομένων, ενδιαφέρουσες γνώσεις, μοτίβα και πληροφορίες, υψηλού επιπέδου, τα οποία μπορούν να χρησιμοποιηθούν με διάφορους τρόπους και να εφαρμοστούν στην λήψη αποφάσεων και στην διαχείριση πληροφοριών. Ως εκ τούτου το data mining θεωρείται ένα από τα πιο σημαντικά εργαλεία για ανάλυση βάσεων δεδομένων και ένα από τα πιο ανερχόμενα στο τομέα της τεχνολογίας πληροφοριών. [14]

1.2 Μηχανές εκμάθησης (machine learning)

Οι μηχανές εκμάθησης είναι ένας τομέας της τεχνητής νοημοσύνης που ασχολείται με αλγόριθμους για αυτοματοποιημένη ανάλυση δεδομένων. Οι αλγόριθμοι αυτοί έχουν την ικανότητα να βελτιώνονται και να προσαρμόζονται από μόνοι τους, έχουν την ικανότητα, δηλαδή να «μαθαίνουν».



ΣΧΗΜΑ 1-2:Machine learning

Η βασική ιδέα των μηχανών εκμάθησης είναι οι αλγόριθμοι εκμάθησης να εκπαιδεύονται και να μαθαίνουν από τα δεδομένα και την εμπειρία τους και να μπορούν να λειτουργούν και να προσαρμόζονται σε διάφορα είδη δεδομένων σε διαφορετικό περιβάλλον εργασίας έτσι ώστε να έχουν την δυνατότητα να αντεπεξέρχονται σε νέα προβλήματα. Εν τέλει κατασκευάζουν ένα στατιστικό μοντέλο πρόβλεψης αποτελούμενο από κάποιες παραμέτρους οι οποίες ορίζονται με βάση την εμπειρία από τα δεδομένα.

1.2.1 Εφαρμογές και παραδείγματα

- Μια εφαρμογή των μηχανών εκμάθησης είναι η ανάλυση του καλαθιού ψωνίσματος (basket analysis), για παράδειγμα όταν έχουμε μια αλυσίδα πολυκαταστημάτων. Σκοπός είναι η εύρεση συσχέτισης μεταξύ των προϊόντων που αγοράζουν οι πελάτες. Για παράδειγμα αν ένας πελάτης που αγοράζει το Χ προϊόν συνήθως αγοράζει και το Υ, και ένας άλλος πελάτης που αγοράζει το Χ δεν αγοράζει το Υ, τότε ο δεύτερος είναι υποψήφιος αγοραστής του Χ προϊόντος. Στόχος είναι η εύρεση αυτών των πελάτων και η τοποθέτηση τους ως cross selling (π.χ η τοποθέτηση των προϊόντων σε κοντινές αποστάσεις, διαφημίσεις κ.α). Με την ίδια λογική θα μπορούσα για κάποιο χρήστη μιας συγκεκριμένης ιστοσελίδας να προβλέψουμε ποιο σύνδεσμο (link) θέλει να επισκεφθεί και να κατεβάσουμε από πριν τα δεδομένα του συνδέσμου για γρηγορότερη πρόσβαση.

- Ένα χρηματοπιστωτικό ίδρυμα, για παράδειγμα μια τράπεζα, προσφέρει δάνεια σε κάποιο πελάτη τα οποία πρέπει να επιστραφούν πίσω με τόκο, συνήθως σε δόσεις. Είναι σημαντικό για την τράπεζα να βρίσκεται σε θέση να προβλέψει εκ των προτέρων τον κίνδυνο ο πελάτης να μην πληρώσει όλο το ποσό του δανείου. Με αυτήν την εκ των προτέρων γνώση η τράπεζα εξασφαλίζει το κέρδος της. Η τράπεζα υπολογίζει τον κίνδυνο αυτό βασιζόμενη στο ποσό του δανείου και σε πληροφορίες που έχει σχετικά με τον πελάτη (επάγγελμα, εισόδημα, αποταμιεύσεις, ηλικία, εξασφαλίσεις κ.α) καθώς και από προηγούμενη εμπειρία της σε εξοφλήσεις δανείων. Σκοπός των μηχανών εκμάθησης είναι η κατασκευή ενός ισχυρού κανόνα που μοντελοποιεί τα δεδομένα του παρελθόντος και τα χαρακτηριστικά του πελάτη και είναι σε θέση να προβλέψει τον κίνδυνο για ένα νέο δάνειο.

- Αναγνώριση χειρόγραφων κειμένων. Για παράδειγμα θέλουμε να αναγνωρίσουμε τη γράφει πάνω σε μια επιταγή ή σε μια επιστολή ή γενικά σε κάποιο χειρόγραφο. Επειδή κάθε άνθρωπος έχει διαφορετικό γραφικό χαρακτήρα, σκοπός των μηχανών εκμάθησης είναι η κατασκευή ενός κανόνα που θα αναγνωρίζει πιο γράμμα αντιστοιχεί σε κάθε χειρόγραφο χαρακτήρα. Για παράδειγμα ο τρόπος γραφής του γράμματος «Ε» διαφέρει από άτομο σε άτομο,

οπότε θέλουμε να βρούμε όλα αυτά τα χαρακτηριστικά που κάνουν όλα τα «Ε» (από διαφορετικούς γραφείς) να είναι όμοια.

- Αναγνώριση προσώπου. Εισάγουμε μια εικόνα ενός προσώπου και οι μηχανές εκπαίδευσης μαθαίνουν να συνδέουν το πρόσωπο της εικόνας με μια εκ το ταυτοτήτων που έχουμε στην βάση δεδομένων.

- Ιατρικές διαγνώσεις. Στην περίπτωση των ιατρικών διαγνώσεων, οι είσοδοι είναι οι πληροφορίες που έχουμε για τον ασθενή : ηλικία, φύλο, ιατρικό ιστορικό, συμπτώματα. Οι μηχανές εκμάθησης συνδέουν τον ασθενή, μοντελοποιόντας τα χαρακτηριστικά του, με μια ασθένεια από την βάση δεδομένων μας.

- Οι μηχανές εκμάθησης χρησιμοποιούνται και για την εύρεση ακραίων τιμών στα δεδομένα μας. Βρίσκουν τιμές οι οποίες δεν υπακούν στον κανόνα που κατασκευάστηκε. Οι τιμές αυτές είναι ανωμαλίες που χρήζουν προσοχής. Για παράδειγμα κάποια απάτη ή παρανομία όπως ξέπλυμα μαύρου χρήματος κ.α.

Όπως γίνεται κατανοητό και από τα πάρα πάνω παραδείγματα η εκμάθηση μπορεί να διαχωριστεί σε δύο σημαντικές υποκατηγορίες, αυτήν που χρειάζεται να προβλέψουμε ένα ενδεχόμενο(π.χ δάνειο σε τράπεζα) και σε αυτές που απλά εξετάζουμε κρυμμένες δομές στα δεδομένα μας (π.χ πολυκατάστημα).[8]

1.3 Εκμάθηση με επίβλεψη και εκμάθηση χωρίς επίβλεψη

1.3.1 Εκμάθηση με επίβλεψη (supervised learning)

Η εκμάθηση με επίβλεψη είναι μια λειτουργεία των μηχανών εκμάθηση στην περίπτωση της οποίας κάθε δείγμα μας αποτελείται από ένα ζευγάρι που περιέχει την επεξηγηματική μεταβλητή *x* και μια «καρτέλα» (label) ,τιμή απόκρισης, *y*. Ο αλγόριθμος εκμάθησης με επίβλεψη αναλύει ένα σύνολο δεδομένων που καλείται σύνολο εκπαίδευσης και παράγει μια συνάρτηση *f* η οποία μπορεί να χρησιμοποιηθεί για να χαρτογραφήσει (ταξινομήσει) νέα δεδομένα. Με άλλα λόγια η *f* καθορίζει την κλάση άγνωστων δεδομένων.

Για να γίνει πιο κατανοητή η διαδικασία παραθέτουμε κάποιους ορισμούς:

Σύνολο εκπαίδευσης: το σύνολο εκπαίδευσης αποτελείται από γνωστά ζευγάρια δεδομένων (x_i, y_i) όπου τα x_i είναι οι επεξηγηματικές μεταβλητές (τα χαρακτηριστικά των δεδομένων) και τα y_i είναι οι κλάσεις που ανήκουν τα αντίστοιχα x_i (η κατηγορία στην οποία ανήκει το αντίστοιχο x_i). Το σύνολο εκπαίδευσης δίνεται σαν είσοδος στον αλγόριθμο εκμάθηση έτσι ώστε ο αλγόριθμος μας να εκπαιδευτεί , να «μάθει» δηλαδή από αυτό το σύνολο δεδομένων.

Ταξινόμηση: στις μηχανές εκμάθησης η ταξινόμηση είναι το πρόβλημα του προσδιορισμού της κλάσης (κατηγορίας) που ανήκει μια νέα παρατήρηση π.χ ένα e-mail είναι spam ή όχι

Σύνολο ελέγχου: το σύνολο ελέγχου (test set) αποτελείται και πάλι από γνωστά ζευγάρια δεδομένων (x_i, y_i). Το σύνολο αυτό χρησιμοποιείται για να ελέγξουμε την απόδοση του μοντέλου μας.

Για την αντιμετώπιση ενός προβλήματος με εκμάθηση με επίβλεψη πρέπει να κάνουμε τα ακόλουθα:

- Καθορίζουμε το είδος των δειγμάτων που έχουμε.
- Συγκεντρώνουμε το σύνολο εκπαίδευσης
- Καθορίζουμε το είδος του βασικού αλγορίθμου (π.χ decision trees)
- Εκπαιδεύουμε τον αλγόριθμο στο σύνολο εκπαίδευσης
- Ελέγχουμε την ακρίβεια του μοντέλου στο σύνολο ελέγχου

Ο αλγόριθμος εκμάθησης με επίβλεψη λειτουργεί ως εξής:

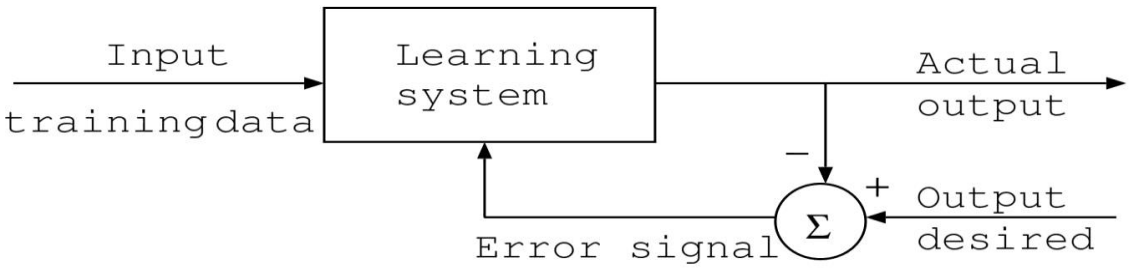
- Δίνουμε ένα σύνολο εκπαίδευσης S που περιέχει m ζευγάρια παραδειγμάτων:

$$S = \{(x_1, y_1), (x_2, y_2), \dots \dots, (x_m, y_m)\}$$

- Ο αλγόριθμος βρίσκει μια συνάρτηση $h: X \to Y$ όπου X είναι ο χώρος εισόδου και Y ο χώρος εξόδου. Η h καλείται συνάρτηση πρόβλεψης

Οι μέθοδοι για μάθηση με επίβλεψη μπορούν να αντιμετωπίσουν προβλήματα ταξινόμησης και παλινδρόμησης και συνήθως αυτές οι μέθοδοι είναι γρήγορες και ακριβείς.

Παραδείγματα αλγορίθμων μάθησης με επίβλεψη είναι τα δέντρα αποφάσεων(decision trees), bagging, boosting, random forest, K – nearest – neighbor (K - NN), γραμμική παλινδρόμηση, λογιστική παλινδρόμηση, μηχανές διανυσματικής υποστήριξης (Support Vector Machines SVM). Στη παρούσα εργασία θα ασχοληθούμε εκτενώς με τις μεθόδους bagging, boosting και SVM.



ΣΧΗΜΑ 1-3: Supervised learning

1.3.2 Εκμάθηση χωρίς επίβλεψη

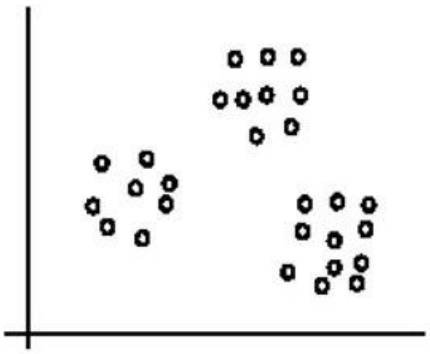
Στις μηχανές εκμάθησης η εκμάθηση χωρίς επίβλεψη αναφέρεται στα προβλήματα όπου τα δεδομένα μας δεν έχουν «καρτέλα» ,τιμή απόκρισης y (unlabeled data). Έτσι η εκμάθηση χωρίς επίβλεψη προσπαθεί να βρει κριμένα μοτίβα ή συσχέτιση μεταξύ των δεδομένων μας. Δηλαδή στην περίπτωση της εκμάθησης χωρίς επίβλεψη δίνουμε σαν είσοδο στον αλγόριθμο μας μόνο τις μεταβλητές $\{x_1, x_2, \dots, x_n\}$. Οι είσοδοι αυτοί για παράδειγμα μπορεί να αντιστοιχούν στα pixels μιας φωτογραφικής, στα αντικείμενα ενός καλαθιού ψωνίσματος κ.α. Η εκμάθηση χωρίς επίβλεψη συνδέεται με την εκτίμηση από ποια κατανομή προέρχονται τα δεδομένα μας.

Παραδείγματα αυτών των μεθόδων είναι τα clustering και πιο συγκεκριμένα ο αλγόριθμος K-means

1.3.2.1 Clustering (ομαδοποίηση)

Το clustering είναι μια τεχνική που χρησιμοποιείται για την εύρεση όμοιων ομάδων δεδομένων που ονομάζονται clusters.

Για παράδειγμα τα δεδομένα μας χωρίζονται σε clusters (Σχήμα 1-4) π.χ για την αρχειοθέτηση εγγράφων



ΣΧΗΜΑ 1-4: Clusters

Το clustering είναι μια από τις πιο διαδεδομένες τεχνικές για μάθηση χωρίς επίβλεψη και χρησιμοποιείται σε πάρα πολλούς τομείς: ιατρική, κοινωνιολογία, βιολογία, αρχαιολογία, ασφάλιση, βιβλιοθήκες κ.α

Ο αλγόριθμος του clustering ή το clustering ομαδοποιεί τα δεδομένα κυρίως βασιζόμενος στην απόσταση που έχουν μεταξύ τους. Μια ομάδα μπορεί να περιγραφεί σε μεγάλο βαθμό από την μέγιστη απόσταση που απαιτείται για να συνδέσουμε τα στοιχεία της ομάδας. Η αναπαράσταση του clustering μπορεί να γίνει χρησιμοποιώντας δενδρογράμματα.

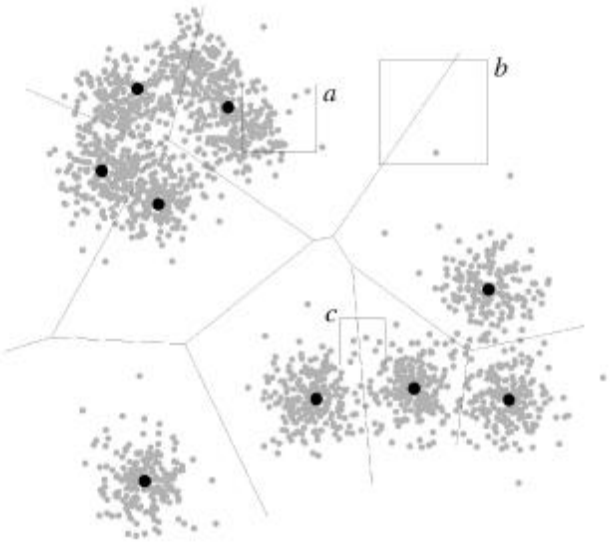
Το clustering δεν είναι πολύ αποτελεσματικό απέναντι σε ακραίες τιμές. Οι τιμές αυτές μπορεί να προκαλέσουν την δημιουργία περισσότερων ομάδων. Παρακάτω παρουσιάζεται ο αλγόριθμος του καθώς και μια σχηματική αναπαράσταση. [8][14][25][34]

Αλγόριθμος k-mean clustering

- Δώσε ένα σύνολο από παρατηρήσεις $(x_1, \dots \dots, x_n) \quad x_i \epsilon R^d$
- Διαχώρισε τις n παρατηρήσεις σε k σύνολα $S = \{s_1, \dots . . s_k\}$ $(k \leq n)$ έτσι ώστε να ελαχιστοποιείται η μέση τετραγωνική απόσταση:

$$min \sum_{i=1}^{k} \sum_{x_j \epsilon S_i} \left|\left|x_j - \mu_i\right|\right|^2$$

Όπου μ_i ο μέσος όρος των σημείων στο σύνολο S_i



ΣΧΗΜΑ 1-5: k-means clustering

2. Μηχανές διανυσματικής υποστήριξης (Support Vector Machines SVM)

Οι μηχανές διανυσματικής υποστήριξης είναι μέθοδος με επίβλεψη η οποία αναλύει δεδομένα και αναγνωρίζει μοτίβα και χρησιμοποιείται για ταξινόμηση δεδομένων καθώς και για παλινδρόμηση.

Οι μηχανές διανυσματικής υποστήριξης δημιουργήθηκαν σαν ιδέα από την Cortes και τον Vapnik το 1993.

Η γενική ιδέα των μηχανών διανυσματικής υποστήριξης είναι ο υπολογισμός ενός υπερεπιπέδου διαχωρισμού μεγιστοποιώντας έτσι τα περιθώρια (αποστάσεις) μεταξύ των κλάσεων των δεδομένων.

Οι μηχανές διανυσματικής υποστήριξης θεωρούνται ως μια από τις πιο αποτελεσματικές μεθόδους για ταξινόμηση και μοντελοποίηση δεδομένων. Εντούτοις παρά τις υψηλές της επιδόσεις η μέθοδος αύτη έχει και κάποιους περιορισμούς. Ειδικότερα οι επιδόσεις της για προβλήματα με πιο πολλές από δυο κλάσεις δεν συγκρίνονται με αυτές που έχει για δυο κλάσεις (δυαδικό πρόβλημα ταξινόμησης). Αυτό συμβαίνει επειδή για να μειώσει τη πεπλεξιμότητα χρησιμοποιεί έναν προσεγγιστικό αλγόριθμο, αυτό έχει σαν συνέπεια την μείωση της απόδοσης της ταξινόμησης.

2.1 Θεωρητικό υπόβαθρο

2.1.1 Γραμμικά διαχωρίσιμες κλάσεις

Οι μηχανές διανυσματικής υποστήριξης αρχικά αντιμετωπίζουν ένα πρόβλημα δύο κλάσεων μεγιστοποιώντας την απόσταση μεταξύ των πλησιέστερων σημείων των δύο κλάσεων. Αυτό μας δίνει δύο πλεονεκτήματα το ότι έχουμε μοναδική λύση του προβλήματος εύρεσης διαχωριστικού υπερεπιπέδου και το ότι μεγιστοποιώντας τα περιθώρια στο σύνολο εκπαίδευσης μπορούμε να πετύχουμε καλύτερη απόδοση της ταξινόμησης.

Υποθέτουμε ότι έχουμε ένα σύνολο από *N* ζευγάρια δειγμάτων : $\{x_i, y_i\}\ \ i = 1, \ldots . N$

$y_i \in \{-1, +1\}$

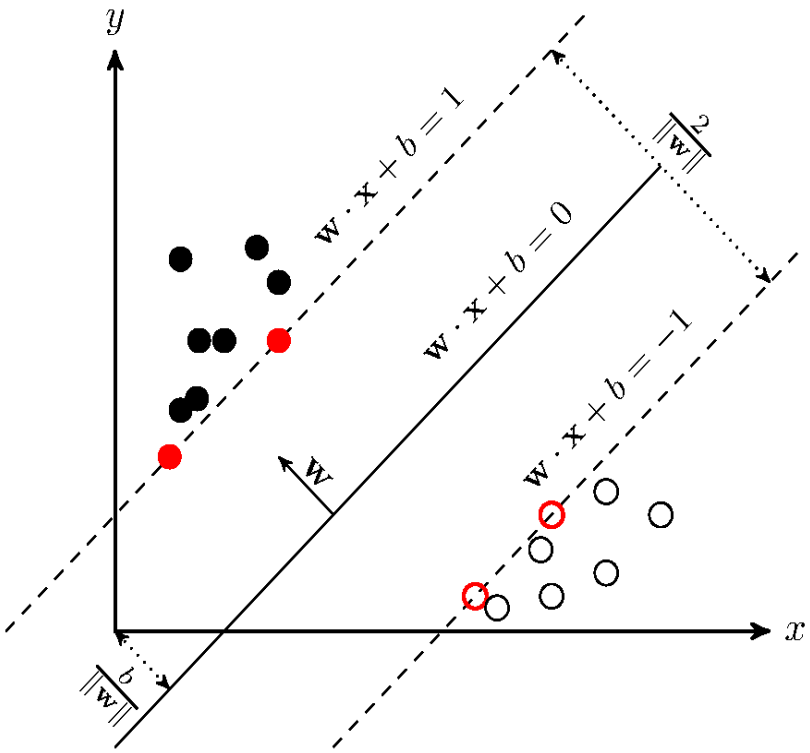
$x_i \in R^p$ το x_i είναι p-διάστασης πραγματικό διάνυσμα.

Θέλουμε μια συνάρτηση της μορφής $f(x)$ που θα υπολογίζει το *y* στο *x*

$$f(x) = b + w^\tau x = b + \sum_{i=1}^{p} w_i x_i$$

Όπου το $b + w^\tau x = 0$ είναι το υπερεπίπεδο μας.

Το *w* είναι διάνυσμα κάθετο στο υπερεπίπεδο.



ΣΧΗΜΑ 2-1: Υπερεπίπεδο μεταξύ δύο κλάσεων 1

Τα διανύσματα υποστήριξης (support vectors) είναι τα σημεία που βρίσκονται πλησιέστερα στο υπερεπίπεδο.

Όπως βλέπουμε και στο Σχήμα 2-1 η μέθοδος επιλέγει τέτοια *b* και *w* έτσι ώστε τα δεδομένα να μπορούν να περιγραφούν από τις παρακάτω ανισώσεις:

- $b + w^{\tau} x_i \leq -1$ για $y_i = -1$
- $b + w^{\tau} x_i \geq +1$ για $y_i = +1$

Ισοδύναμα:

$y_i(b + w^{\tau} x_i) \geq +1 \Rightarrow y_i(b + w^{\tau} x_i) - 1 \geq 0$

Τα διανύσματα υποστήριξης μπορούν να περιγραφούν από δυο υπερεπίπεδα (όπως φαίνεται και στο Σχήμα 1) τα:

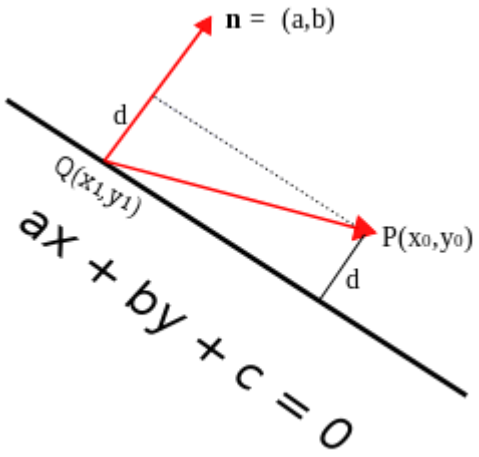
- $b + w^{\tau} x_i = -1$ Ε1

- $b + w^{\tau}x_i = +1$ E2

Η απόσταση μεταξύ των δύο υπερεπιπέδων είναι ίση με $\frac{2}{||w||}$

<u>Απόδειξη</u>

Απόσταση σημείου από ευθεία



ΣΧΗΜΑ 2-2: Αποστασή σημείου από ευθεία

$$d = \frac{QP\ n}{||n||}$$

$$QP = (x_0 - x_1, y_0 - y_1)$$

$$QP\ n = a(x_0 - x_1) + b(y_0 - y_1)$$

$$||n|| = \sqrt{a^2 + b^2}$$

$$d = \frac{|a(x_0 - x_1) + b(y_0 - y_1)|}{\sqrt{a^2 + b^2}}$$

$$c = -ax_1 - by_1$$

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

Έστω (x,y) ένα διανύσματα υποστήριξης

$$ax + by + 1 = 0$$

$by = -1 - ax$

$y = \frac{-1-ax}{b}$

Για x=0 $y = \frac{-1}{b}$

(0,-1/b) ϵ Ε1

Αντικαθιστώντας στο $d = \frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$

Έχουμε:

$d = \frac{|b\frac{-1}{b}-1|}{\sqrt{a^2+b^2}}$

Οπότε $d = \frac{2}{||w||}$

Όπως προ είπαμε στόχος του SVM είναι η μεγιστοποίηση αυτής της απόστασης.

Οπότε έχουμε το ακόλουθο πρόβλημα μεγιστοποίησης:

$$\max \frac{2}{||w||}$$

Με τον περιορισμό:

$$y_i(b + w^\tau x_i) - 1 \geq 0 \quad (1)$$

Ή ισοδύναμα:

$$\min \frac{||w||}{2} \;\; subject\ to: y_i(b + w^\tau x_i) - 1 \geq 0$$

Για λόγους απλότητας στην εκτέλεση του προγραμματισμού θα αντικαταστήσουμε το ||w|| με $||w||^2$ δηλαδή καταλήγουμε στο πρόβλημα:

$$\min \frac{||w||^2}{2} \;\; subject\ to: y_i(b + w^\tau x_i) - 1 \geq 0$$

Το πρόβλημα αυτό μπορεί να λυθεί εισάγοντας τους πολλαπλασιαστές Lagrange ***α*** και ακολούθως ελαχιστοποιώντας της συνάρτηση Lagrange:

$$L_p = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} a[y_i(x_i w + b) - 1] \;\; a \geq 0 \quad (2)$$

Στην συνέχεια υπολογίζουμε τις μερικές παραγώγους της L_p ως προς *w* και *b* και τις θέτουμε ίσες με μηδεν:

$$\frac{\partial L_p}{\partial w} = w - \sum_{i=1}^{N} a_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{N} a_i y_i x_i \qquad (3)$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{N} a_i y_i = 0 \qquad (4)$$

Αντικαθιστώντας τις (3) και (4) στην (2) έχουμε:

$L_D = \sum_{i=1}^{N} a_i - \sum_{i,j=1}^{N} a_i a_j y_i y_j x_i x_j \;\; subject\ to\ a_i \geq 0 \;\; \sum_{i=1}^{N} a_i y_i = 0$

$a_i > 0$ αν το x_i είναι διάνυσμα υποστήριξης α_i=0 αν το x_i δεν είναι διάνυσμα υποστήριξης.

Η λύση του προβλήματος μπορεί να βρεθεί μεγιστοποιώντας την L_D δηλαδή καταλήγουμε στο ακόλουθο πρόβλημα:

$\max L_D = \sum_{i=1}^{N} a_i - \sum_{i,j=1}^{N} a_i a_j y_i y_j x_i x_j \;\; subject\ to\ a_i \geq 0 \;\; \sum_{i=1}^{N} a_i y_i = 0$

Λύνοντας το πρόβλημα μεγιστοποίησης υπολογίζουμε το α_i και έτσι χρησιμοποιώντας την (3) βρίσκουμε και το *w* ενώ το *b* το βρίσκουμε χρησιμοποιώντας την (1).

Έτσι καταλήγουμε στην συνάρτηση υπερεπιπέδου για ταξινόμηση καινούργιων στοιχείων:

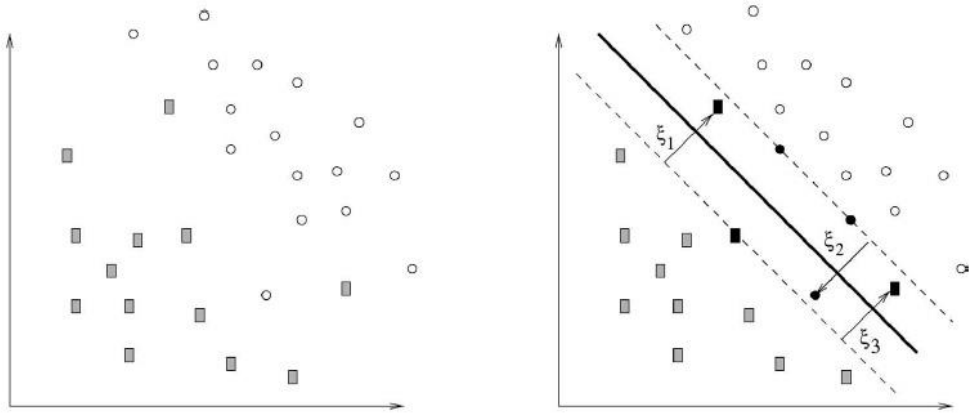
$$f(x) = b + w^{\tau} x$$

$$G(x) = sign\ f(x)$$

Η λύση αυτή λειτουργεί στην περίπτωση που οι κλάσεις μας είναι τέλεια διαχωρισμένες και έτσι ένα γραμμικό υπερεπίπεδο μπορεί να μας δώσει την βέλτιστη λύση. Στην περίπτωση των μη γραμμικά διαχωρίσιμων δεδομένων χρειάζεται μια μη γραμμική λύση.

2.1.2 Μη γραμμικά διαχωρίσιμες κλάσεις

Σε αρκετά προβλήματα οι κλάσεις των δεδομένων μας μπορεί να μην είναι γραμμικά διαχωρίσιμες. Για την αντιμετώπιση αυτών των προβλημάτων τα SVM εισάγουν ένα σύνολο από «χαλαρές» μεταβλητές $\xi = \{\xi_1, \xi_2, \ldots., \xi_N\}$ για τα σημεία που βρίσκονται στην λάθος πλευρά του περιθωρίου. Με αυτό τον τρόπο πετυχαίνουμε μια μερική χαλάρωση των περιορισμών μας (Σχήμα 2-3).



ΣΧΗΜΑ 2-3: Χαλαρές μεταβλητές

Υποθέτουμε και πάλι ότι έχουμε ένα σύνολο από N ζευγάρια δειγμάτων : $\{x_i, y_i\}\ i = 1, \ldots . N$

$y_i \in \{-1, +1\}$

$x_i \in R^p$

Σκοπός και πάλι είναι η μεγιστοποίηση του περιθώριο λαμβάνοντα υπόψιν τις χαλαρές μεταβλητές και τροποποιώντας τους περιορισμούς από:

- $b + w^\tau x_i \leq -1$ για $y_i = -1$
- $b + w^\tau x_i \geq +1$ για $y_i = +1$

Σε:

- $b + w^\tau x_i \leq -1 + \xi_i$ για $y_i = -1$
- $b + w^\tau x_i \geq +1 - \xi_i$ για $y_i = +1$

$\xi_i \geq 0$

Ισοδύναμα:

$$y_i(b + w^\tau x_i) \geq +1 - \xi_i \Rightarrow\ y_i(b + w^\tau x_i) - 1 + \xi_i \geq 0 \qquad \xi_i \geq 0 \qquad (5)$$

Όπου η ανισότητα (5) αντιπροσωπεύει την ποσότητα του κατά πόσο η πρόβλεψη f(x) θα βρίσκεται στην λάθος πλευρά του περιθωρίου. Ως εκ τούτου ορίζουμε ένα άνω φράγμα για το συνολική ποσότητα $\sum_{i=1}^{N} \xi_i$. Έτσι ψάχνουμε για ένα υπερεπίπεδο διαχωρισμού με όσο το δυνατό μεγαλύτερο περιθώριο ελέγχοντας ,όμως, το ποσοστό των προβλέψεων που θα βρίσκονται στην λάθος πλευρά του περιθωρίου να είναι φραγμένο από ένα όριο.

Λαμβάνοντας υπόψιν τις χαλαρές μεταβλητές έχουμε το παρακάτω πρόβλημα ελαχιστοποίησης:

$$\min \frac{||w||^2}{2} \;\; subject\ to: \begin{cases} y_i(b + w^\tau x_i) \geq +1 - \xi_i \\ \xi_i \geq 0 \\ \sum_{i=1}^{N} \xi_i \leq K \end{cases}$$

Μπορούμε να ξαναγράψουμε το πρόβλημα ως:

$$\min\left(\frac{||w||^2}{2} + C\sum_{i=1}^{N} \xi_i\right) subject\ to: \begin{cases} y_i(b + w^\tau x_i) \geq +1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

Όπου η σταθερά *K* αντικαταστάθηκε με μια παράμετρο κόστους *C.*

Όπως και πριν το πρόβλημα μπορεί να λυθεί προσαρμόζοντας τους πολλαπλασιαστές Lagrange:

$$L_p = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} a_i[y_i(x_i w + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i \quad (6)$$

Υπολογίζουμε τις μερικές παραγώγους ως προς *w*, *b* κα *ξ*$_i$ και τις θέτουμε ίσες με μηδέν:

$$\frac{\partial L_p}{\partial w} = w - \sum_{i=1}^{N} a_i y_i x_i = 0 \;\Rightarrow w = \sum_{i=1}^{N} a_i y_i x_i \quad (7)$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{N} a_i y_i = 0 \quad (8)$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \sum_{i=1}^{N} a_i - \sum_{i=1}^{N} x_i = 0 \;\Rightarrow C = \sum_{i=1}^{N} a_i - \sum_{i=1}^{N} x_i \quad (9)$$

Αντικαθιστώντας την (7) και (8) στην (6) καταλήγουμε στην νέα μορφή:

$$L_D = \sum_{i=1}^{N} a_i - \sum_{i,j=1}^{N} a_i a_j y_i y_j x_i x_j$$

$$0 \leq a_i \leq C \quad \sum_{i=1}^{N} a_i y_i = 0$$

Δηλαδή καταλήγουμε στο ακόλουθο πρόβλημα μεγιστοποίησης

$$\max\left(L_D = \sum_{i=1}^{N} a_i - \sum_{i,j=1}^{N} a_i a_j y_i y_j x_i x_j\right) subject\ to: \begin{cases} 0 \le a_i \le C \\ \sum_{i=1}^{N} a_i y_i = 0 \end{cases}$$

Το *w* υπολογίζεται από την σχέση (7) από όλα τα μη μηδενικά α_i (α_i >0 αν το x_i είναι διάνυσμα υποστήριξης, 0 αλλιώς). Το *b* μπορεί να υπολογιστεί χρησιμοποιώντας τα διανύσματα υποστήριξης. Και πάλι η συνάρτηση πρόβλεψης ορίζεται ως:

$$G(x) = sing\ f(x).$$

Η παράμετρος κόστους *C* μπορεί να ρυθμιστεί έτσι ώστε τα «μαλακό» περιθώριο να περιλαμβάνει έναν συγκεκριμένο αριθμό από παρατηρήσεις *i*. Αν η παράμετρος κόστους *C* είναι πολύ μεγάλη η λύση θα οδηγήσει σε Over-fitting.

2.2 Θεωρία πυρήνων για SVM

Μέχρι τώρα οι μηχανές διανυσματικής υποστήριξης στόχευαν στην εύρεση ενός γραμμικού δεσμού μεταξύ των χώρων (κλάσεων) που θέλουμε να διαχωρίσουμε. Αυτό δεν είναι πρακτικά χρήσιμο όταν ο διαχωρισμός είναι μη γραμμικός. Ωστόσο, η παραπάνω προσέγγιση μπορεί εύκολα να γενικοποιηθεί έτσι ώστε να δημιουργεί μη γραμμικούς δεσμούς μεταξύ των χώρων. Αυτό επιτυγχάνεται με την μεταφορά του προβλήματος μας από τον αρχικό μας χώρο σε ένα χώρο Hilbert υψηλότερης διάστασης (Σχήμα 2-4). (ο χώρος θα πρέπει να είναι Hilbert θα πρέπει να ορίζονται εσωτερικά γινόμενα).

Η μεταφορά αυτή γίνεται μέσω μιας απεικόνισης:

$$\varphi(x): X^m \to X^n \quad n \ge m$$

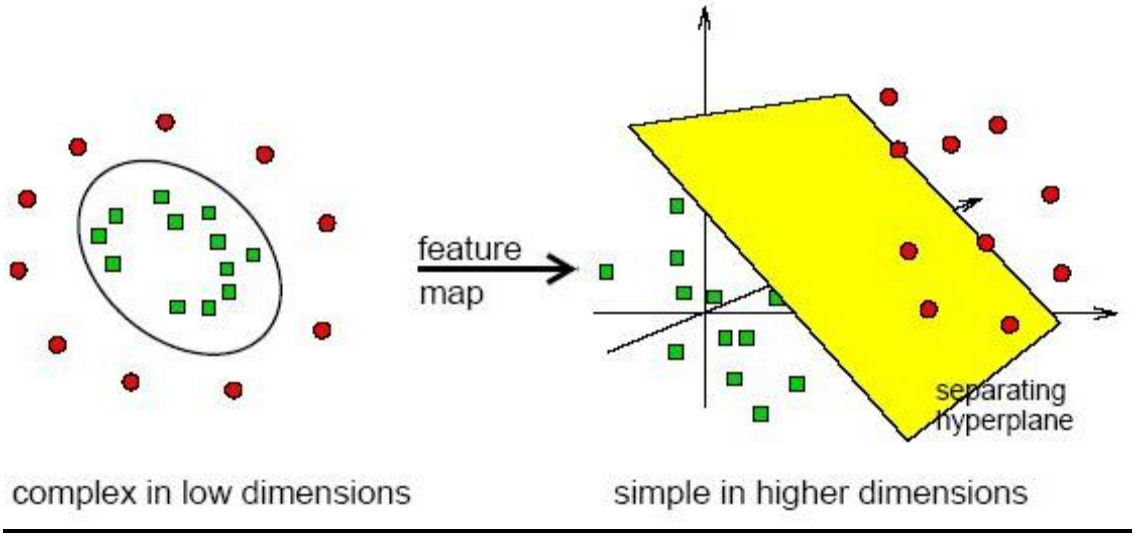
Οπότε τώρα η συνάρτηση μας που θα υπολογίζει το y στο x θα πάρει την μορφή:

$$f(x) = b + w^{\tau}\varphi(\chi)$$

Και το πρόβλημα μας θα γίνει:

$$\max\left(\sum_{i=1}^{N} a_i - \sum_{i,j=1}^{N} a_i a_j y_i y_j \langle \varphi(x_i), \varphi(x_j) \rangle\right) subject\ to: \begin{cases} 0 \le a_i \le C \\ \sum_{i=1}^{N} a_i y_i = 0 \end{cases}$$

Όπου $\langle x, y \rangle = x^T y$



ΣΧΗΜΑ 2-4: Διαχωρισμός σε υψηλότερη διάσταση

Η $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$ ονομάζεται συνάρτηση πυρήνα. Ο παραπάνω κανόνας ταξινόμησης εξαρτάται μόνο από την συνάρτηση πυρήνα. Οπότε σε αυτή τη μέθοδο χρειάζεται μόνο ο υπολογισμός του εσωτερικού γινομένου.[21][10][30][5][35][27]

Παραθέτουμε τις πιο δημοφιλής (για SVM) συναρτήσεις πυρήνα στον παρακάτω πίνακα:

ΠΙΝΑΚΑΣ 2-1:KERNEL FUNCTIONS

$\varphi(x)$	$K(\varphi(x_i)\varphi(x_j))$
Degree d polynomial	$(1 + \langle x_i, x'_j \rangle)^d$
Radial basis	$\exp\{\frac{-\|\|x_i - x_J\|\|^2}{\sigma}\}$
Neural network	$\tanh(\theta_1 \langle x_i, x'_j \rangle + \theta_2)$

2.3 Παράδειγμα εφαρμογή μηχανών διανυσματικής υποστήριξης (Svm)

Υποθέτουμε ότι θέλουμε να δημιουργήσουμε ένα φίλτρο email το οποίο θα μπορεί να διακρίνει τα spam emails από τα μη-spam. Τα spam είναι e-mails που αποστέλλονται στους παραλήπτες με σκοπό να διαφημίσουν διάφορα προϊόντα, ιστοσελίδες κ.α, καθώς και να ξεγελάσουν το παραλήπτη αποσπώντας του χρήματα.

Το σύνολο δεδομένων μας ονομάζεται spam και το βρίσκουμε στην βιβλιοθήκη “kernlab” της R. Το σύνολο δεδομένων συλλέχθηκε από τα εργαστήρια της Hewllet-Packard και αποτελείται από 4601 e-mails που ταξινομούνται ως spam και non-spam. Επιπρόσθετα η ταξινόμηση γίνεται με βάση 57 μεταβλητές οι οποίες αντιπροσωπεύουν την συχνότητα εμφάνισης συγκεκριμένων λέξεων και χαρακτήρων στα e-mails.

Το σύνολο δεδομένων περιέχει 2788 e-mails ταξινομημένα ως non-spam και 1812 ταξινομημένα ως spam.

Καταρχήν εγκαθιστούμε στην R το πακέτο “e1071” και “kernlab” τα οποία θα χρησιμοποιήσουμε για την επεξεργασία των δεδομένων μας. Στην συνέχεια καλούμε τις βιβλιοθήκες των αντίστοιχών πακέτων:

```
install.packages('e1071',dependencies=TRUE)
library(e1071)
install.packages('kernlab',dependencies=TRUE)
library("kernlab")
```

Τα δεδομένα μας βρίσκονται στο πακέτο “kernlab” και έχουμε πρόσβαση σε αυτά με την ακόλουθη εντολή:

```
data(spam)
```

Ακολούθως χωρίζουμε το σύνολο δεδομένων μας σε σύνολο εκπαίδευσης και σύνολο δοκιμής. Το σύνολο δοκιμής αποτελείται από το 30% των στοιχείων του αρχικού σύνολο δεδομένων επιλεγμένων τυχαία ενώ το σύνολο εκπαίδευσης από το υπόλοιπο 70% των στοιχείων του σύνολο δεδομένων μας:

```
index <- 1:nrow(spam)
testindex <- sample(index, trunc(length(index)*30/100))
testset <- spam[testindex,]
trainset <- spam[-testindex,]
```

εκπαιδεύουμε το μοντέλο μας στο σύνολο εκπαίδευσης με την εντολή svm() του πακέτου “e1071” . Για συνάρτηση πυρήνα χρησιμοποιούμε την “radial”

```
> model <- svm(type ~ ., data = trainset, method = "C-
classification", kernel = "radial", cost = 10, gamma = 0.1)
```

Με την εντολή summary() η R μας δίνει τα αποτέλεσματα της εκπαίδευσης καθώς και άλλες πληροφορίες για την προσαρμογή των μηχανών διανυσματικής υποστήριξης:

```
summary(model)

Call:

svm(formula = type ~ ., data = trainset, method = "C-
classification",

    kernel = "radial", cost = 10, gamma = 0.1)

Parameters:

   SVM-Type:  C-classification

 SVM-Kernel:  radial

       cost:  10

      gamma:  0.1

Number of Support Vectors:  1632

 ( 688 944 )


Number of Classes:  2

Levels:

 nonspam spam
```

Από την εντολή summary() παρατηρούμε ότι ο αριθμός των διανυσμάτων υποστήριξης είναι 1632 (688 για την μια κλάση και 944 για την άλλη), ακόμη παρατηρούμε ότι τα δεδομένα μας αποτελούνται από της κλάσεις nonspam και spam.

Στην συνέχεια χρησιμοποιώντας το σύνολο δοκιμής ελέγχουμε την απόδοση του ταξινομητή μας με την εντολή predict(.,.). Με την εντολή αυτή το μοντέλο μας ταξινομεί τα στοιχεία του σύνολο δοκιμής σε spam και nonspam σύμφωνα με την «εμπειρία» που απέκτησε από το σύνολο εκπαίδευσης.

```
pred <- predict(model, testset)
```

Με την εντολή table() δημιουργούμε έναν πίνακα ο οποίος μας δείχνει ποια emails ταξινομήθηκαν σωστά. Ακριβέστερα μας δείχνει πόσα από τα emails που ταξινομήθηκαν ως nonspam είναι πράγματι nonspam ,πόσα που ταξινομήθηκαν ως nonspam είναι spam, πόσα που ταξινομήθηκαν σαν spam είναι nonspam και πόσα που ταξινομήθηκαν σαν spam είναι πράγματι spam.

```
acc <- table(pred, testset$type)
```

```
  pred      nonspam spam
  nonspam     829   79
  spam         27  445
```

όπως παρατηρούμε 829 emails ταξινομήθηκαν ορθώς ως nonspam και 445 emails ταξινομήθηκαν ορθώς ως spam. Αντιθέτως 79 emails ταξινομήθηκαν ως nonspam ενώ ήταν spam και 27 ταξινομήθηκαν ως spam ενώ ήταν nonspam.

Τέλος με την εντολή classAgreement() παρουσιάζεται το ποσοστό επιτυχίας της ταξινόμησης βάση του προηγούμενου πίνακα.

```
classAgreement(acc)
```

```
$diag
[1] 0.9072464
$kappa
[1] 0.8027779
$rand
[1] 0.8315772
$crand
[1] 0.661991
```

το diag συμβολίζει το ποσοστό των στοιχείων που βρίσκονται στην κύρια διαγώνιο του πίνακα μας, δηλαδή το ποσοστό των σωστά ταξινομημένων στοιχείων.

Το kappa συμβολίζει την διορθωμένη τιμή του diag, η διόρθωση κρίνεται απαραίτητη για τον λόγο ότι το μέγεθος των δύο κλάσεων είναι διαφορετικό.

Παρατηρούμε ότι η τιμή της diag είναι 0.9072464 και της kappa 0.8027779 δηλαδή ο ταξινομητή μας έχει 80.28% επιτυχία.[11][28][4]

3. BOOSTING και BAGGING

3.1 BOOSTING

Η βασική ιδέα του Boosting είναι η ανάπτυξη ενός αδύναμου ταξινομητή, σταδιακά.

Ο αδύνατος ταξινομητής πρέπει να είναι τουλάχιστον καλύτερος από την τυχαία επιλογή. Δηλαδή για έναν δυαδικό ταξινομητή η υπόθεση θα πρέπει να είναι σωστή με πιθανότητα >0,5

Το σύνολο εκπαίδευσης επιλέγεται βασιζόμενο στις προηγούμενες επιδόσεις του ταξινομητή. Ακριβέστερα ο αλγόριθμος του boosting εκπαιδεύει τον «αδύναμο» ταξινομητή σε πολλούς γύρους χρησιμοποιώντας «βάρη» στο αρχικό σύνολο εκπαίδευσης.

Ο αλγόριθμος του boosting εκπαιδεύει τον αρχικό ταξινομητή με ίσα βάρη για όλα τα δεδομένα του αρχικού σύνολο εκπαίδευσης, μετά εκπαιδεύει τους υπόλοιπους ταξινομητές ανακατανέμοντας τα βάρη στο σύνολο εκπαίδευσης με την εξής λογική: Σε κάθε γύρο τα βάρη των λάθος ταξινομημένων παραδειγμάτων αυξάνονται κάνοντας έτσι τον ταξινομητή να επικεντρωθεί στα λάθος ταξινομημένα παραδείγματα.

Ο σκοπός του αλγορίθμου είναι να βρει μια υπόθεση $H_t(x) : X \to \{-1, +1\}$

Για παράδειγμα αν η i-οστή παρατήρηση προβλέφθηκε λάθος από την $h_t(x)$ τότε το βάρος της $w_i(t+1)$ αυξάνεται για την $h_{t+1}(x)$. Έτσι ο επόμενος ταξινομητής θα επιχειρήσει να διορθώσει τα λάθη του προηγούμενου.

Τέλος η συνολική (τελική) πρόβλεψη θα είναι γραμμικός συνδυασμός από τις $h_t(x)$

$$\mathrm{H(x)} = \mathrm{sign}\ \sum_{t=1}^{T} a_t h_t(x)$$

Ένα παράδειγμα για την καλύτερη κατανόηση του Boosting είναι το εξής: υποθέτουμε ότι θέλουμε να δημιουργήσουμε ένα φίλτρο email το οποίο θα μπορεί να διακρίνει τα spam emails από τα μη-spam. Η προσέγγιση του προβλήματος μας θα ήταν η ακόλουθη: ξεκινούμε συλλέγοντας όσα πιο πολλά παραδείγματα μπορούμε και από spam και από μη-spam emails. Στη συνέχεια δημιουργούμε ένα σύνολο με όλα τα παραδείγματα μαζί με ετικέτες (labels) που αναγράφουν εάν είναι spam ή όχι και εκπαιδεύουμε τον αλγόριθμο μας έτσι ώστε να ταξινομεί τα emails σε spam και μη-spam. Λαμβάνοντας ένα νέο email ο αλγόριθμος μας θα επιχειρήσει να προβλέψει εάν είναι spam ή όχι. Ο στόχος ,δηλαδή, είναι η δημιουργία ενός κανόνα που θα καθιστά την πρόβλεψη ακριβέστερη για νέα emails.

Η δημιουργία ενός εξαιρετικά ακριβούς κανόνα πρόβλεψης είναι δύσκολη. Αντιθέτως δεν είναι δύσκολη η δημιουργία πολλών «μετρίως» ακριβών κανόνων π.χ αν εμφανίζεται η φράση «buy now» το email θα προβλέπεται σαν spam. Ο σκοπός του

boosting είναι αυτός ακριβώς, η εύρεση δηλαδή, πολλών τέτοιων κανόνων και στο τέλος η σύνθεση τους για την δημιουργία ενός ταξινομητή υψηλής ακρίβειας.

Υπάρχουν διάφοροι boosting αλγόριθμοι όπως ο ADABOOST, ο ADABOOSTM1, ο ADABOOSTM2, ADABOOSTMH και Arc-x4. Αναλυτικότερα :

- ADABOOST είναι κατασκευασμένος για δυαδικής ταξινόμησης προβλήματα
- ADABOOST.M1 είναι κατασκευασμένος για ταξινόμηση σε περισσότερες από 2 κλάσεις. Δουλεύει αν ο «αδύναμος» ταξινομητής μπορεί να πετύχει 50% ακρίβεια
- ADABOOST.MH δημιουργεί ένα σύνολο από δυαδικά προβλήματα δηλαδή δημιουργεί και επιλύει το πρόβλημα εάν για ένα x είναι σωστό το συγκεκριμένο y ή ένα από τα υπόλοιπα y.
- ADABOOST.M2 δημιουργεί ένα σύνολο από δυαδικά προβλήματα δηλαδή δημιουργεί και επιλύει το πρόβλημα εάν για ένα x είναι σωστό το συγκεκριμένο y ή ένα y'
- ADABOOST.R είναι κατασκευασμένος για μοντέλα παλινδρόμησης.
- Arc-x4 διαφέρει σε δύο σημεία από τον adaboost. Πρώτο στο ότι τα βάρη στην t επανάληψη υπολογίζονται βάση του ποσοστού των φορών m_i της λάθος ταξινόμησης του δείγματος από τους προηγούμενους $t-1$ ταξινομητές. Δεύτερο το ότι ο τελική ταξινόμηση εκλέγεται με πλειοψηφική απόφαση.

Περιγραφή του αλγορίθμου Adaboost

Ο αλγόριθμος παίρνει για είσοδο το σύνολο εκπαίδευσης $(x_1,y_1),\ldots.(x_m,y_m)$ όπου κάθε x_iανήκει σε κάποιο χώρο X και κάθε y_i σε κάποιο χώρο Y. Για δυαδικά προβλήματα ο χώρος Y ορίζεται ως ο χώρος $\{-1,+1\}$. Ο αλγόριθμος καλεί τον δοθέν βασικό αλγόριθμο εκπαίδευσης σε διαδοχικούς γύρους t = 1,. . . .,T. Ο αλγόριθμος κατανέμει τα βάρη ισόνομα στο σύνολο εκπαίδευσης. Η κατανομή των βαρών για το i στοιχείο στον γύρο t συμβολίζεται ως $D_t(i)$. Όπως είπαμε και πιο πριν τα βάρη των λάθος ταξινομημένων στοιχείων αυξάνεται έτσι ο βασικός αλγόριθμος εκπαίδευσης επικεντρώνεται σε αυτά.

Η δουλειά του βασικού αλγόριθμου εκπαίδευσης είναι να ελαχιστοποιήσει το σφάλμα

$$\varepsilon_t = P_{r_i \sim d_t}[h_t(x_i) \neq y_i]$$

Όταν πάρουμε τον βασικό ταξινομητή ο AdaBoost αλγόριθμος επιλέγει μια παράμετρο $a_t \in R$ η οποία μέτρα την σημαντικότητα του h_t. Για δυαδικά προβλήματα η παράμετρος αυτή ορίζεται ως:

$$a_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right).$$

Στη συνέχεια η κατανομή D_t ενημερώνεται όπως φαίνεται στον αλγόριθμο. Τέλος ο τελικός ταξινομητής ορίζεται ως η πλειοψηφική απόφαση των T βασικών ταξινομητών όπου το α_t είναι το βάρος προσαρμοσμένο στον h_t. [18][20][27][32]

Αλγόριθμος ADABOOST

Είσοδος: $(x_1,y_1),\ldots.(x_m,y_m)$ όπου $x_i \in X$, $y_i \in Y = \{-1,+1\}$

Αρχικοποίησε $D_1(i) = \frac{1}{m}$

Για t= 1, ..., T:

- Εκπαίδευσε τον βασικό αλγόριθμο εκμάθησης χρησιμοποιώντας την κατανομή D_t
- Πάρε τον βασικό ταξινομητή $h_t : X \rightarrow R$
- Επέλεξε $\alpha_t \in R$
- Ενημέρωσε :

$$D_{t+1}(i) = \frac{D_t(i)\exp(-a_t y_i h_t(x_i))}{Z_t}$$

Όπου Z_t είναι ο όρος κανονικοποίησης (επιλεγμένος έτσι ώστε η D_{t+1} να είναι κατανομή)

Δώσε το τελικό ταξινομητή:

$$H(x) = sign(\sum_{t=1}^{T} a_t h_t(x))$$

Πιο κάτω παραθέτουμε τους αλγόριθμους Adaboost.M1 και Adaboost.M2

Αλγόριθμος: Adaboost.M1

Είσοδος: σύνολο εκπαίδευσης $S = \{(x_i, y_i), \dots \dots, (x_n, y_n)\}$, $x_i \epsilon X$, $y_i \epsilon C = \{c_1, c_2, \dots \dots, c_m\}$, T: ο αριθμός των επαναλήψεων, I: ο βασικός αλγόριθμος εκμάθησης.

Έξοδος: ο Boosted ταξινομήτης:

$$H(x) = \arg max_{y \epsilon C} \sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right) [h_t(x) = y]$$

- $D_1(i) = \frac{1}{N}$ $\gamma\iota\alpha\ i = 1,2, \dots \dots, N$
- **Για t=1 μέχρι Τ κάνε**:

$$h_t = I(S, D_t)$$

$$\varepsilon_t = \sum_{i=1}^{N} D_t(i)\,[h_t(x_i) \neq y_i]$$

Αν $\varepsilon_t > 0.5$ τότε:

$$T = t - 1$$

Δώσε αποτέλεσμα

Τέλος το αν.

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$$

$$D_{t+1}(i) = D_t(i)\ \beta_t^{\,1-[h_t(x_i) \neq y_i]} \quad \gamma\iota\alpha\ i = 1, \dots \dots N$$

Κανονικοποίησε το $D_{t+1}(i)$ ώστε να αποτελεί κατανομή
Τέλος το για.

Adaboost.M2

Είσοδος: σύνολο εκπαίδευσης $S = \{(x_i, y_i), \dots\dots, (x_n, y_n)\}$, $x_i \epsilon X$, $y_i \epsilon C = \{c_1, c_2, \dots\dots, c_m\}$, T: ο αριθμός των επαναλήψεων, I: ο βασικός αλγόριθμος εκμάθησης.

Έξοδος: ο Boosted ταξινομητής:

$$H(x) = \arg max_{y \epsilon C} \sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right) h_t(x, y) \quad , h_t(x, y) \epsilon [0,1]$$

- $D_1(i) = \frac{1}{N}$ $\gamma\iota\alpha\ i = 1,2, \dots\dots, N$
- $w_{i,y}^1 = {}^{D(i)}/_{m-1}$ για i=1,……,N $y \epsilon C - \{y_i\}$
- Για t=1 μέχρι T κάνε:

$$W_i^t = \sum_{y \neq y_i} w_{i,y}^t$$

$$g_t(i, y) = \frac{w_{i,y}^t}{W_i^t} \quad \gamma\iota\alpha\ y \neq y_i$$

$$D_t(i) = \frac{W_i^t}{\sum_{i=1}^{N} W_i^t}$$

$$h_t = I(S, D_t)$$

$$\varepsilon_t = \frac{1}{2} \sum_{i=1}^{N} D_t(i) \left(1 - h_t(x_i, y_i) + \sum_{i, y \neq y_i} q_t(i, y) h_t(x_i, y)\right)$$

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$$

$$w_{i,y}^{t+1} = w_{i,y}^t \beta_t^{\frac{1}{2}(1 + h_t(x_i, y_i) - h_t(x_i, y))} \quad , i = 1, .. N\ y \epsilon C - \{y_i\}$$

Τέλος το για.

3.1.1 Σφάλματα

3.1.1.1 Σφάλμα εκπαίδευσης

Το σφάλμα εκπαίδευσης είναι ο λόγος των λάθος ταξινομημένων στοιχείων προς τον αριθμό όλων των στοιχείων στο σύνολο εκπαίδευσης. Ειδικότερα οι Schapire και Singer απέδειξαν ότι το σφάλμα εκπαίδευσης του τελικού ταξινομητή είναι φραγμένο ως εξής:

$$\left(\frac{1}{m}\right)|i: H\{x_i \neq y_i\}| \leq \left(\frac{1}{m}\right)\sum_{i=1}^{m}\exp(-y_i f(x_i)) = \prod_{t=1}^{T} Z_t \qquad (1)$$

Όπου $f(x) = \sum_{t=1}^{T} a_t h_t(x)$

Και $H(x) = sign\, f(x)$

Αυτό συμπεραίνεται από το ότι

$\exp\left(-y_i f(x_i)\right) \geq 1 \quad \alpha\nu\, y_i \neq H(x_i)$

Η σχέση (1) μας υποδεικνύει ότι το σφάλμα μπορεί να μειωθεί επιλέγοντας κατάλληλα a_t και h_t σε κάθε γύρο έτσι ώστε να ελαχιστοποιείται η σχέση:

$$Z_t = \sum_{i=1}^{m} D_t(i)\exp(-a_t y_i\, h_t(x_i))$$

Στην περίπτωση που έχουμε δυαδικό ταξινομητή αυτό μας οδηγεί στην επιλογή του a_t ως:

$$a_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon t}{\varepsilon t}\right)$$

και έτσι παίρνουμε το φράγμα για το σφάλμα εκπαίδευσης

$$\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T}\left(2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right) = \prod_{t=1}^{T}\sqrt{1-4{\gamma_t}^2} \leq \exp\left(-2\sum_{t=1}^{T}{\gamma_t}^2\right)$$

Όπου ορίζουμε σαν $\gamma_t = ½ - \varepsilon_t$. (Αυτή η σχέση αποδείχτηκε αρχικά από τον Freund και τον Schapire.)

$\varepsilon_t = ½ - \gamma_t$

Εξ υποθέσεως η πρόβλεψη μιας τυχαίας μεταβλητής έχει σφάλμα ½ (στην περίπτωση των δυαδικών προβλημάτων) οπότε η γ_t είναι το μέτρο για το πόσο καλύτερη είναι η πρόβλεψή μας h_t από την τυχαία πρόβλεψη. Αυτό οδηγεί στο συμπέρασμα ότι κάθε

βασικός ταξινομητής που είναι ελαφρώς καλύτερος από την τυχαία ταξινόμηση έτσι ώστε $\gamma_t \geq \gamma$ για κάποιο $\gamma \geq 0$ οδηγεί σε εκθετική μείωση του σφάλματος εκπαίδευσης, αυτό φαίνεται από τον όρο $\exp(-2\sum_{t=1}^{T} {\gamma_t}^2)$. Έτσι αποδεικνύεται ότι ο αλγόριθμος Adaboost είναι πράγματι ένας αλγόριθμος που μπορεί να μετατρέψει έναν αδύναμο αλγόριθμο εκμάθησης σε ισχυρό αλγόριθμο που ταξινομεί τα στοιχεία μας με χαμηλό σφάλμα.

3.1.1.2 Γενικό σφάλμα

Το γενικό σφάλμα ορίζεται ως η πιθανότητα της λάθος ταξινόμησης ενός νέου δείγματος, ενώ το σφάλμα ελέγχου είναι ο λόγος των λάθος ταξινομημένων δειγμάτων προς όλα τα δείγματα στο σύνολο δοκιμής. Έτσι αναμένουμε το γενικό σφάλμα και το σφάλμα ελέγχου να είναι περίπου ίσα.

Οι Freund και Schapire έδειξαν πως φράσσύνολοαι το γενικό σφάλμα του τελικού ταξινομητή με έναν όρο που περιλαμβάνει το σφάλμα εκπαίδευσης, το μέγεθος m του δείγματος, την VC-διάσταση του χώρου του βασικού ταξινομητή και τον αριθμό των επαναλήψεων T του boosting. Το γενικό σφάλμα είναι:

$$P_r[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

Όπου ως P_r [.] συμβολίζουμε εμπειρικά την πιθανότητα λάθος ταξινόμησης στο σύνολο εκπαίδευσης. Αυτή η σχέση δείχνει ότι το boosting θα κορεστεί (overfit) αν τρέξει για πάρα πολλές επαναλήψεις δηλαδή αν το T γίνει αρκετά μεγάλο. Πολλά πειράματα, όμως, έδειξαν ότι αυτό δεν συμβαίνει ακόμη και όταν ο αλγόριθμος τρέξει για χιλιάδες επαναλήψεις. Αυτό οδήγησε σε μια ενναλακτική προσέγγιση του προβλήματος, την θεωρία περιθωρίων (margin theory).

Το περιθώριο ενός δείγματος (x,y) ορίζεται ως:

$$margin_f(x,y) = \frac{y\, f(x)}{\sum_t^T |a_t|} = \frac{y \sum_t^T a_t h_t(x)}{\sum_t^T |a_t|}$$

Αυτή η ποσότητα είναι θετική μόνο και μόνο αν ο ταξινομητής H ταξινομήσει σωστά το δείγμα. Επιπλέον το μέγεθος των περιθωρίων μπορεί να ερμηνευθεί ως μέτρο εμπιστοσύνης της πρόβλεψης. Έχει αποδειχθεί ότι μεγαλύτερα περιθώρια στο σύνολο εκπαίδευσης οδηγούν σε μεγαλύτερο άνω φράγμα για το γενικό σφάλμα

$$P_r[margin_f(x,y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

Για κάθε θ > 0. Παρατηρούμε ότι αύτη η σχέση είναι ανεξάρτητη από τον αριθμό των επαναλήψεων T. Ακόμη το Boosting είναι ιδιαίτερα αποτελεσματικό στην μείωση των περιθωρίων, αυτό συμβαίνει γιατί το Boosting επικεντρώνεται στα δείγματα με τα μικρότερα περιθώρια. [22][31][32]

3.1.2 Εφαρμογή - παράδειγμα Boosting

Στο παρόν παράδειγμα θα επιχειρήσουμε να αντιμετωπίσουμε με την βοήθεια του Boosting ένα πρόβλημα 2 κλάσεων. Το σύνολο δεδομένων μας αποτελείται από 286 μετρήσεις από ανθρώπους που είχαν θεραπευτεί από καρκίνο του στήθους και οι οποίοι χωρίζονται σε δύο κλάσεις αυτούς που έπαθαν υποτροπή και σε αυτούς που δεν έπαθαν υποτροπή. Η μια κλάση περιέχει 201 στοιχεία και η άλλη 85. Τα στοιχεία περιγράφονται από 9 μεταβλητές.

Αντιμετώπιση του προβλήματος μας στην R:

Καταρχήν εγκαθιστούμε το πακέτο “adabag” και φορτώνουμε τις βιβλιοθήκες “adabag”, “rpart” και “mlbench” στην R με τις ακόλουθες εντολές:

```
install.packages("adabag")
library(adabag)
library(rpart)
library(mlbench)
```

στην συνέχεια φορτώνουμε τα δεδομένα “BreastCancer”:

```
data(BreastCancer)
```

χωρίζουμε το σύνολο δεδομένων μας σε σύνολο εκπαίδευσης και σύνολο δοκιμής

```
l <- length(BreastCancer[,1])
train <- sample(1:l,2*l/3)
```

Τρέχουμε τον αλγόριθμο adaboost στο σύνολο εκπαίδευσης:

```
Canceradaboost <- boosting(Class ~.,data=BreastCancer[train,-1],mfinal=25,control=rpart.control(maxdepth=3))
```

Ελέγχουμε την ακρίβεια της πρόβλεψης του αλγορίθμου μας στο σύνολο δοκιμής:

```
Canceradaboost.pred <- predict.boosting(Canceradaboost,newdata=BreastCancer[-train,-1], newmfinal=15)

Canceradaboost.pred$confusion
```

```
                Observed Class
Predicted Class benign malignant
      benign       148         7
      malignant      4        74
```

με την εντολή “Canceradaboost.pred$confusion” παίρνουμε το πιο πάνω πίνακα στον οποίο παρουσιάζονται ποια στοιχεία του σύνολο δοκιμής ταξινομήθηκαν σωστά και ποια λάθος. Όπως παρατηρούμε 148 στοιχεία του σύνολο δοκιμής ταξινομήθηκαν σωστά ως benign ενώ 7 στοιχεία ταξινομήθηκαν λανθασμένα ως benign. 74 στοιχεία ταξινομήθηκαν ορθώς ως malignant και τέλος 4 στοιχεία ταξινομήθηκαν ως “malignant” αντί για “bening”.

Με την εντολή “Canceradaboost.pred$error” παίρνουμε το σφάλμα πρόβλεψης

Canceradaboost.pred$error

```
[1] 0.0472103
```

[28][26][17][7].

3.2 Bagging

Το bootstrap aggregating ή αλλιώς bagging είναι ένας μετά-αλγόριθμος που έχει ως σκοπό την βελτίωση ενός ταξινομητή καθώς και μοντέλων παλινδρόμησης.

Η γενική ιδέα του bagging είναι απλή και ελκυστική. Δημιουργούμε bootstrap αντίγραφα από το σύνολο εκπαίδευσης, κάθε ταξινομητής εκπαιδεύεται στο συγκεκριμένο σύνολο. Στο τέλος κάθε ταξινομητής προβλέπει μια μεταβλητή εξόδου για κάθε το διάνυσμα εισόδου και το αποτέλεσμα λαμβάνεται με πλειοψηφική απόφαση, δηλαδή το η τιμή με τις περισσότερες ψήφους επιλέγεται ως η μεταβλητή απόκρισης.

Αλγόριθμος BAGGING

Είσοδος: S:σύνολο εκπαίδευσης, T: αριθμός επαναλήψεων, N: bootstrap δείγμα, I: βασικός ταξινομητής

Έξοδος: Bagged ταξινομητής: $H(x) = sign(\sum_{t=1}^{T} h_t(x))$ όπου $h_t(x) \ \epsilon [-1, +1]$

- Για t=1 μέχρι T εκτέλεσε:

  S_t ←δειγματοληψία με επανατοποθέτηση (N,S)
  h_t ←I(S_t)

Τέλος

Καταρχήν δημιουργούμε bootstrap δείγματα από το αρχικό σύνολο εκπαίδευσης. Ένα bootstrap δείγμα δημιουργείται από *N'* δείγματα επιλεγμένα με δειγματοληψία με επανατοποθέτηση από το αρχικό σύνολο εκπαιδεύσεως που περιέχει *N* δείγματα (*N'*≤*N*). Έτσι δημιουργούμε T bootstrap δείγματα S_t $(t = 1, \ldots, T)$και ο βασικός ταξινομητής εκπαιδεύεται σε κάθε bootstrap. Ένας τελικός ταξινομητής *H* κατασκευάζεται και μας δίνει την πρόβλεψη της κλάσης ενός στοιχείου συνήθως με πλειοψηφική απόφαση με τις ισοπαλίες να επιλέγονται αυθαίρετα.

Λόγο του ότι χρησιμοποιείται δειγματοληψία με επανατοποθέτηση κάποιες παρατηρήσεις μπορεί να επαναλαμβάνονται σε κάθε T_i. Η πιθανότητα να επιλεγεί ένα στοιχείο τουλάχιστο μια φορά είναι $1-(1-\frac{1}{N})^N$. για μεγάλο N και N'=N η πιθανότητα γίνεται $1-{}^1/_e$ οδηγώντας έτσι στο να έχουμε 63,3% μοναδικά δείγματα στο σύνολο T_i.

Το bagging έχει αποδειχθεί ότι μπορεί να προσφέρει σημαντική βελτίωση στην ακρίβεια. Ενώ επισημαίνεται ότι η σταθερότητα της μεθόδου πρόβλεψης είναι το κλειδί για την επιτυχία του Bagging.

Παράδειγμα Majority voting. Στον πίνακα 1 φαίνεται απλοποιημένα η διαδικασία της πλειοψηφικής ψηφοφορίας. Ο classifier 1 και ο classifier 2 ψηφίζουν ότι η παρατήρηση ανήκει στην κλάση 1 ενώ ο classifier 3 ψηφίζει ότι η παρατήρηση ανήκει στη κλάση 2. Στο τέλος ο αλγόριθμος μας αποφασίζει σύμφωνα με πλειοψηφική απόφαση ότι η παρατήρηση ανήκει στην κλάση 1

ΠΙΝΑΚΑΣ 3-1: MAJORITY VOTING

	CLASS 1 VOTE	CLASS 2 VOTE
Classifier 1	1	0
Classifier 2	1	0
Classifier 3	0	1
BAGGING	**2**	1

Αν και το bagging χρησιμοποιείται συνήθως με decision tree ως βασικό ταξινομητή εντούτοις μπορεί να χρησιμοποιηθεί με οποιοδήποτε τύπο ταξινομητή ή μοντέλων.

Αν ο βασικός ταξινομητής μας είναι ασταθής (unstable), δηλαδή αν μικρές αλλαγές στο σύνολο εκπαίδευσης οδηγούν σε μεγάλες αλλαγές στην απόφαση που θα μας δώσει ο ταξινομητής, τότε το Bagging βελτιώνει σημαντικά την ακρίβεια της πρόβλεψης. Τώρα αν ο βασικός ταξινομητής μας είναι σταθερός (stable) τότε το bagging σε κάποιες περιπτώσεις μπορεί και να μειώσει την ακρίβεια της πρόβλεψης. Αυτό συμβαίνει για τον λόγο ότι ο βασικός ταξινομητής εκπαιδεύεται σε μικρότερο σύνολο δεδομένων, αφού το μέγεθος του σύνολο εκπαίδευσης μειώνεται με το bagging. Παραδείγματα ασταθών ταξινομητών είναι τα νευρωνικά δίκτυα και τα δέντρα αποφάσεων και παλινδρόμησης. Ενώ παραδείγματα σταθερών ταξινομητών είναι τα SVM και οι K-nearest neighbor μέθοδοι.

Πλεονεκτήματα του bagging είναι η μείωση της συνδιακύμανσης του μοντέλου εκπαίδευσης καθώς και η αποφυγή overfitting (το overfitting είναι το φαινόμενο κατά το οποίο το στατιστικό μας μοντέλο αρχίζει να περιγράφει το «θόρυβο» το τυχαίο λάθος ,δηλαδή, αντί την σχέση μεταξύ των μεταβλητών μας. φαινόμενα overfitting παρατηρούνται κυρίως όταν το μοντέλο μας είναι πολύ περίπλοκο όπως για παράδειγμα όταν περιλαμβάνει πολλές παραμέτρους. Ένα overfit μοντέλο έχει πολύ χαμηλή απόδοση πρόβλεψης).[18][6][21]

3.2.1 Δέντρα αποφάσεων

Όπως έχουμε πει το bagging χρησιμοποιεί συνήθως δέντρα αποφάσεων ως βασικό ταξινομητή, για την ακριβή κατανόηση της μεθόδου πιο κάτω παρουσιάζεται η λειτουργία και το μαθηματικό υπόβαθρο πάνω στο οποίο είναι χτισμένες οι μέθοδοι βασισμένες σε δέντρα.
Οι Tree based μέθοδοι είναι μέθοδοι που χρησιμοποιούνται στην εξόρυξη δεδομένων και στην μηχανική εκμάθηση, χρησιμοποιούνται ως μοντέλα παλινδρόμησης και ταξινόμησης. Δηλαδή όπως και στις υπόλοιπές μεθόδους πρόβλεψης στοχεύουμε σε μια εκτιμήτρια συνάρτηση $f(x)$. Σε αυτές τις μεθόδους η $f(x)$ προσεγγίζεται από ένα γραμμικό συνδυασμό από υψηλής τάξης συναρτήσεις διχοτόμησης:

$$I(x_j < t_k) \text{ ή } I(x_j > t_k)$$

όπου το x_j είναι το i-οστό στοιχείο του διανύσματος x και όπου t_k είναι το σημείο διχοτόμησης.

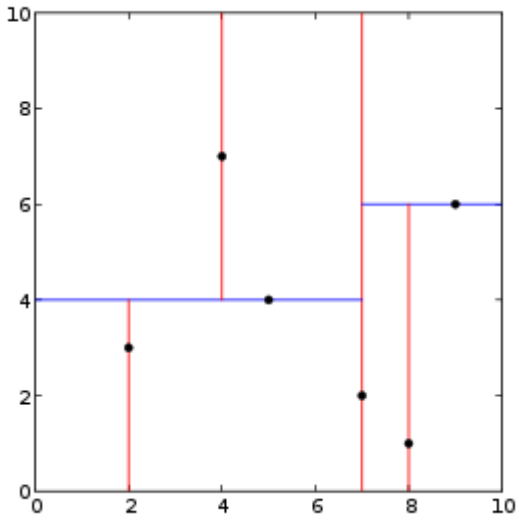
Καταρχήν ξεκινούμε με όλα τα αρχικά δεδομένα, θέλουμε να κάνουμε διαχωρισμό στο j-οστό στοιχείο του διανύσματος εισόδου X και να ορίσουμε το σημείο διχοτόμησης s έτσι ώστε να ελαχιστοποιείται η συνάρτηση:

$$min \sum_{i=1}^{n} (Y_i - c_1)^2 I(X_{ij} \leq s) + min \sum_{i=1}^{n} (Y_i - c_2)^2 I(X_{ij} > s)$$

Με άλλα λόγια ψάχνουμε για το βέλτιστο στοιχείο και το βέλτιστο σημείο διχοτόμησης έτσι ώστε το μέσο τετραγωνικό σφάλμα να ελαχιστοποιείται. Λαμβάνοντας τον βέλτιστο συνδυασμό διχοτόμησης δημιουργούνται ορθογώνια χωρία:

$$\{x: I(x_j \leq s)\} \text{ και } \{x: I(x_j > s)\}$$

Συνεχίζουμε την ίδια μεθοδολογία για m βήματα οπότε πετυχαίνουμε μια διαμέριση σε m+1 περιοχές ,σχήμα 3-1:



ΣΧΗΜΑ 3-1: ΔΙΑΜΕΡΙΣΗ ΧΩΡΙΟΥ

Έτσι το δένδρο διακλαδώνεται και μεγαλώνει, υπερβολική διακλάδωση όμως οδηγεί σε φαινόμενα overfitting, οπότε πρέπει να ορίσουμε κάποιο κριτήριο διακοπής. Μια αποτελεσματική στρατηγική «κλαδέματος του δέντρου» είναι να λάβουμε υπόψιν μας μια συνάρτηση κόστους- πολυπλοκότητας. Για παράδειγμα για δέντρο με m κόμβους, όπου κάθε κόμβος στην ουσία αντιστοιχεί σε ένα ορθογώνιο χωρίο, συμβολίζουμε με V_k την μεταβολή μεταξύ των ορθογωνίων και με N_k τον αριθμό των παρατηρήσεων σε αυτό το ορθογώνιο. Η συνάρτηση κόστους ορίζεται ως:

$$\sum_{k=1}^{m} N_k V_k + am$$

Παρατηρούμε ότι όσο αναπτύσσεται το δέντρο ο πρώτος όρος της συνάρτησης κόστους μειώνεται αλλά ο δεύτερος όρος αυξάνεται έτσι ώστε να επιβάλει ποινή στην πολυπλοκότητα του δέντρου.

3.2.2 Δέντρα ταξινόμησης

Η διαφορά στην περίπτωση των δέντρων ταξινόμησης είναι ότι η διακλάδωση επιλέγεται βασιζόμενη στην ελαχιστοποίηση κάποιας συνάρτησης απώλειας, όπως είναι: το σφάλμα ταξινόμησης, ο δείκτης Gini $\sum_{k=1}^{K} \widehat{p_k}(1-\widehat{p_k})$ όπου $\widehat{p_k}$ είναι η πιθανότητα η παρατήρηση να ταξινομηθεί στην κ κλάση, και η εντροπία $\sum_{k=1}^{k} \widehat{p_k}\, log\widehat{p_k}$.

Μια άλλη αποδοτική μέθοδος ταξινόμησης είναι η Random Forest η οποία περιγράφεται στο επόμενο κεφάλαιο.

3.2.2.1 Random forest

Τα Random forests είναι μια σημαντική τροποποίηση του bagging η οποία δημιουργεί ένα μεγάλο σύνολο από ασυσχέτιστα δέντρα και στη συνέχεια παίρνει την απόφαση βασιζόμενη στην αποτελέσματα τον δέντρων αυτών (ουσιαστικά παίρνει το μέσο όρο των αποφάσεων τους). Στις πλείστες των περιπτώσεων η μέθοδος αυτή έχει απόδοση περίπου ίδια με αυτή του bagging με την διαφορά όμως ότι είναι απλούστερη και εκπαιδεύεται πιο εύκολα. Έτσι τα random forests θεωρούνται ιδιαίτερα προσιτές μέθοδοι και χρησιμοποιούνται σε πληθώρα προβλημάτων.

Θεωρητικό υπόβαθρο

Το bagging βασίζεται στην δημιουργία αμερόληπτων μοντέλων και στην μείωση της διασποράς, ενώ λειτουργεί καλύτερα με ασταθείς ταξινομητές. Ως εκ τούτου τα δέντρα αποφάσεων αποτελούν ιδανική επιλογή αφού θεωρούνται ασταθείς ως ταξινομητές και έχουν ,σχετικά, χαμηλή μεροληψία.

Η γενική ιδέα του random forest είναι μείωση της διασποράς του Bagging μειώνοντας την συσχέτιση ρ μεταξύ των δέντρων. Αυτό πετυχαίνετε με την τυχαία επιλογή των μεταβλητών εισόδου. Ακριβέστερα μετά από την επέκταση ενός δέντρου στο bootstrapped σύνολο πριν από κάθε διαχωρισμό επιλέγουμε m μεταβλητές από τις p (p είναι οι μεταβλητές εισόδου) m ≤ p τυχαία ως υποψήφιες προς διαχωρισμό.

Μετά την δημιουργία B τέτοιων δέντρων $T(x, \Theta_b)\ \ b = 1, \ldots, B$ η random forest συνάρτηση πρόβλεψης γίνεται:

$$\widehat{f_f^B}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x, \Theta_b)$$

Όπου η Θ_b χαρακτηρίζει το b δέντρο ως προς τον αριθμό των χωρισμένων μεταβλητών, των αριθμό των διαχωρισμών σε κάθε κόμβο και των τελικό αριθμό των κόμβων. Μειώνοντας το m μειώνεται και η συσχέτιση μεταξύ των δέντρων και ως συνέπεια και η διασπορά του τελικού μέσου όρου.

Αλγόριθμος Random Forest

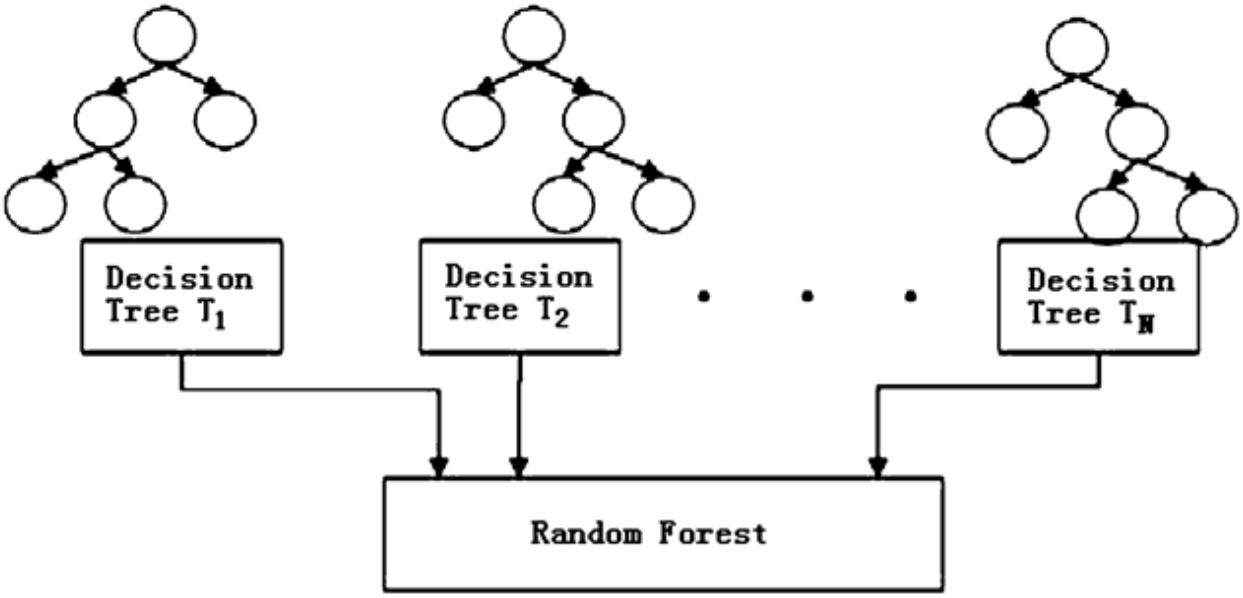
Αλγόριθμος random forest για παλινδρόμηση και ταξινόμηση:

1. Για b=1 μέχρι B:
   1.1. Δημιούργησε ένα bootstrap δείγμα C με μέγεθος N από το σύνολο εκπαίδευσης
   1.2. Κατασκεύασε ένα random forest δέντρο T_b στο bootstrap δείγμα ακολουθώντας τα επόμενα βήματα σε κάθε τερματικό κόμβο του δέντρου μέχρι να επιτευχθεί ο ελάχιστος αριθμός κόμβων n_{min}:
      1.2.1. Επέλεξε m μεταβλητές από τις p τυχαία
      1.2.2. Επέλεξε της βέλτιστες μεταβλητές διαχωρισμού από τις m
      1.2.3. Διαχώρισε τον κόμβο σε δύο άλλους
2. Δώσε το σύνολο των δέντρων $\{T_b\}_1^B$

Για πρόβλεψη σε νέο σημείο x:

$$Παλινδρόμηση: \hat{f}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

$$Ταξινόμηση: \widehat{G_{rf}}^B(x) = majority\ vote\ \{\widehat{C_b}(x)\}_1^B$$



ΣΧΗΜΑ 3-2: Random Forests

3.2.2.2 Ταξινόμηση και παλινδρόμηση

Στην περίπτωση της ταξινόμησης στο random forest κάθε δέντρο δίνει μια κλάση σαν μεταβλητή απόκρισης και ακολούθως με πλειοψηφική απόφαση επιλέγεται σε πια κλάση ανήκει η είσοδος που δώσαμε $\widehat{G_{rf}}^{B}(x) = majority\ vote\ \{\widehat{C_b}(x)\}_1^B$,η προκαθορισμένη τιμή για το m είναι $\sqrt{p}$ και ο ελάχιστος αριθμός κόμβων είναι 1. Στην περίπτωση της παλινδρόμησης απλά παίρνουμε το μέσο όρο των προβλέψεων από κάθε δέντρο $\hat{f}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$ ενώ η προκαθορισμένη τιμή για το m είναι $\frac{P}{3}$ και ο ελάχιστος αριθμός κόμβων είναι 5.

3.2.2.3 Σημαντικότητα μεταβλητών

Τα random forests χρησιμοποιούνται και για να αποφανθούμε για την σημαντικότητα των μεταβλητών μας σε προβλήματα ταξινόμησης και παλινδρόμησης.

Το βασικό κριτήριο για την σημαντικότητα μιας δοθέν μεταβλητής είναι η αύξηση του μέσου τετραγωνικού σφάλματος σε ένα δέντρο από το σύνολο όταν οι παρατηρούμενες τιμές αυτής της μεταβλητής ανταλλάσσονται με τιμές από το σύνολο OOB (out of bag samples). Το σύνολο OOB αποτελείται από τις παρατηρήσεις που δεν χρησιμοποιήθηκαν στην κατασκευή του συγκεκριμένου δέντρου.[27][35]

3.2.3 Εφαρμογή παράδειγμα Bagging

Στην εφαρμογή αυτή θα χρησιμοποιήσουμε το ίδιο σύνολο δεδομένων που χρησιμοποιήσαμε για το boosting. Όπως και πριν το σύνολο δεδομένων μας αποτελείται από 286 μετρήσεις από ανθρώπους που είχαν θεραπευτεί από καρκίνο του στήθους και οι οποίοι χωρίζονται σε δύο κλάσεις αυτούς που έπαθαν υποτροπή και σε αυτούς που δεν έπαθαν υποτροπή. Η μια κλάση περιέχει 201 στοιχεία και η άλλη 85. Τα στοιχεία περιγράφονται από 9 μεταβλητές.

Επεξεργασία στην R:

Φορτώνουμε τις βιβλιοθήκες (adabag, rpart, mlbench) και το σύνολο δεδομένων μας χρησιμοποιώντας τις ακόλουθες εντολές:

```
library(adabag)

library(rpart)

library(mlbench)

data(BreastCancer)
```

χωρίζουμε το σύνολο δεδομένων μας σε σύνολο εκπαίδευσης και σύνολο δοκιμής:

```
l <- length(BreastCancer[,1])

train<- sample(1:l,2*l/3)
```

χρησιμοποιούμε τον αλγόριθμο bagging για επεξεργασία των δεδομένων μας στο σύνολο εκπαίδευσης:

```
Cancerbagging <- bagging(Class ~.,data=BreastCancer[train,-
1],mfinal=20,control=rpart.control(maxdepth=3))
```

Τρέχουμε τον αλγόριθμο στο σύνολο δοκιμής για να ταξινομήσουμε τα στοιχεία του σύνολο μας, έτσι ώστε να ελέγξουμε την ακρίβεια ταξινόμησης:

```
Cancerbagging.pred <-
predict.bagging(Cancerbagging,newdata=BreastCancer[-train,-1])
```

Κατασκευάζουμε το πίνακα στον οποίο παρουσιάζονται τα στοιχεία του σύνολο δοκιμής που ταξινομήθηκαν σωστά και αυτά που ταξινομήθηκαν λανθασμένα.

```
Cancerbagging.pred$confusion

               Observed Class

Predicted Class benign malignant

      benign       144         5

      malignant     10        74
```

όπως παρατηρούμε 144 στοιχεία του σύνολο δοκιμής ταξινομήθηκαν σωστά ως benign ενώ 5 στοιχεία ταξινομήθηκαν λανθασμένα ως benign. 74 στοιχεία ταξινομήθηκαν ορθώς ως malignant και τέλος 10 στοιχεία ταξινομήθηκαν ,λανθασμένα, ως “malignant” αντί για “bening”.

Τέλος η παρακάτω εντολή μας δίνει το σφάλμα πρόβλεψης:

```
Cancerbagging.pred$error

[1] 0.06437768
```

Παρατηρούμε ότι το Boosting ταξινόμησε λανθασμένα 7 στοιχεία ως benign και 4 ως malignant, δηλαδή ταξινόμησε, λανθασμένα, συνολικά 13 στοιχεία ενώ το Bagging ταξινόμησε λανθασμένα 5 ως benign και 10 ως malignant, δηλαδή συνολικά 15 λάθος ταξινομημένα στοιχεία, 2 περισσότερα από το boosting. Τέλος το σφάλμα για το bagging είναι 0.06437768 μεγαλύτερο από αυτό του boosting που είναι 0.0472103.[17][12][28]

4. Bagging και Boosting σε συνδυασμό με μηχανές διανυσματικής υποστήριξης

Έχοντα αναλύσει στα προηγούμενα κεφάλαια τις μεθόδους μηχανών εκμάθησης bagging, boosting και μηχανών διανυσματικής υποστήριξης στο παρών κεφάλαιο θα μελετήσουμε και θα συγκρίνουμε συνδυασμό αυτών των μεθόδων. Ακριβέστερα θα δούμε σε εφαρμογή τις μεθόδους bagging και boosting να χρησιμοποιούν για βασικό ταξινομητή μηχανές διανυσματικής υποστήριξης. Ακολούθως θα παραθέσουμε κάποια αποτελέσματα, από την εφαρμογή των συνδυασμών αυτών σε διάφορα σύνολα δεδομένων και θα συγκρίνουμε την ακρίβεια πρόβλεψης και τα σφάλματα τους.

Ανακεφαλαιώνοντας οι μηχανές διανυσματικής υποστήριξης είναι ταξινομητές που μπορούν να πετύχουν υψηλή ακρίβεια και έχουν καλές ιδιότητες γενίκευσης. Η απόδοση τους εξαρτάται σε μεγάλο βαθμό από τις τιμές που χρησιμοποιούμε για παραμέτρους η εύρεση των οποίων είναι ιδιαίτερα χρονοβόρα όταν το σύνολο δεδομένων μας είναι μεγάλο.

Από την άλλη το bagging και boosting είναι δυο προσεγγίσεις που έχουν ως στόχο την βελτίωση της ακρίβειας πρόβλεψης, χρησιμοποιώντας αδύναμους αλγόριθμος εκμάθησης. Τα bagging και boosting με τις ιδιότητες τους μπορεί να μειώσουν αισθητά τον απαιτούμενο χρόνο υπολογισμού των παραμέτρων για τις μηχανές διανυσματικής υποστήριξη και να βελτιώσουν την ακρίβεια ταξινόμησης.

4.1 Boosting με μηχανές διανυσματικής υποστήριξης (SVM) για βασικό ταξινομητή

Η ιδέα για χρήση των μηχανών διανυσματικής εκμάθησης για βασικό ταξινομητή στον αλγόριθμο boosting αρχικά δεν δείχνει να είναι η ιδανικότερη για το λόγο ότι οι μηχανές διανυσματικής υποστήριξης αποτελούν ταξινομητή που γενικά είναι δύσκολο να εκπαιδευτεί και έτσι η ιδέα αυτή μπορεί να μειώσει την απόδοση του boosting. Παρόλα αυτά χρησιμοποιώντας ως βασικό ταξινομητή τις μηχανές διανυσματικής υποστήριξης με συνάρτηση πυρήνα την Radial basis function $\exp\{\frac{-||x_i-x_j||^2}{\sigma}\}$ ξεκινώντας με μεγάλες τιμές του σ (δηλαδή αδύναμο ταξινομητή) και μειώνοντας σταδιακά το σ σε κάθε επανάληψη, μπορούμε να περιμένουμε καλύτερη απόδοση του Adaboost. Αυτό είναι λογικό γιατί διαφοροποιώντας το σ παίρνουμε ένα σύνολο ταξινομητών μηχανών διανυσματικής υποστήριξης που προσαρμόζεται και γενικεύεται καλύτερα από ότι θα είχαμε αν χρησιμοποιούσαμε σταθερό σ.

4.1.1 Αλγόριθμος AdaboostSVM

Όπως έχουμε πει σκοπός μας είναι η χρησιμοποίηση μηχανών διανυσματικής υποστήριξης με συνάρτηση πυρήνα radial basis ως βασικό ταξινομητή στον αλγόριθμο Adaboost. Το βασικό πρόβλημα μας είναι ο ορισμός της τιμής του σ της radial basis

συνάρτησης των μηχανών διανυσματικής υποστήριξης κατά την διάρκεια των επαναλήψεων. Όπως έχουμε αναφέρει αν κρατήσουμε σταθερή την τιμή του *σ* θα δημιουργηθούν προβλήματα στην απόδοση του αλγορίθμου μας. Αυτό συμβαίνει για τον λόγω του ότι αν ορίσουμε την τιμή του *σ* πολύ μεγάλη τότε ο βασικός ταξινομητής μας θα είναι πολύ αδύναμος με αποτέλεσμα η ακρίβεια του σε πολλές περιπτώσεις να πέφτει κάτω του 50% και έτσι ο αλγόριθμος του Adaboost δεν θα μπορεί να ανταποκριθεί, αφού βασική προϋπόθεση για τον Adaboost είναι ο βασικός ταξινομητής να έχει ακρίβεια τουλάχιστον μεγαλύτερη της τυχαίας επιλογής (>50%). Από την άλλη αν θέσουμε την τιμή του σ αρκετά μικρή τότε ο βασικός ταξινομητής θα γίνει αρκετά ισχυρός, με αποτέλεσμα το σύνολο ταξινομητών που θα κατασκευαστεί να είναι πανομοιότυποι και τα «λάθη» τους στις πλείστες των περιπτώσεων θα είναι κοινά. Επιπλέον η μικρή τιμή του σ πιθανόν να οδηγήσει σε φαινόμενα υπερπροσαρμογής (overfit). Έτσι το boosting θα καταστεί αναποτελεσματικό.

Οπότε καταλήγουμε στο συμπέρασμα ότι κατά την εφαρμογή του boosting με ισχυρό βασικό ταξινομητή ο ταξινομητής αυτός θα πρέπει να αποδυναμωθεί κατάλληλα προκειμένου να επωφεληθεί από το boosting. Ως εκ τούτο στην περίπτωση που θα χρησιμοποιήσουμε μηχανές διανυσματικής υποστήριξης με radial basis συνάρτηση πυρήνα για βασικό ταξινομητή θα προτιμήσουμε μια σχετικά μεγάλη τιμή του *σ*. Η προτεινόμενη μορφή του αλγορίθμου AdaboostSVM φαίνεται στον πιο κάτω αλγόριθμο και περιγράφεται ως εξής: Αρχικά ορίζεται μια μεγάλη τιμή για το *σ* που αντιστοιχεί στον ταξινομητή μηχανών διανυσματικής υποστήριξης με radial basis συνάρτηση πυρήνα, έτσι αποδυναμώνεται ο βασικός μας ταξινομητής. Στη συνέχεια οι radial basis μηχανές διανυσματικής υποστήριξης εκπαιδεύονται σε τόσους γύρους όσους είναι δυνατόν να διατηρήσουν ακρίβεια πάνω από 50%. Αν αυτό δεν είναι εφικτό τότε μειώνουμε ελαφρός την τιμή του *σ* για να αυξήσουμε την ακρίβεια των radial basis μηχανών διανυσματικής υποστήριξης πάνω από 50%. Η μικρή μείωση της τιμής του *σ* έχει σαν αποτέλεσμα οι νέοι ταξινομητές radial basis μηχανές διανυσματικής υποστήριξης να μην γίνουν πολύ ισχυροί, δηλαδή να έχουν μέτρια ακρίβεια. Ο λόγος που χρειαζόμαστε μέτρια ακρίβεια είναι ότι με αυτό τον τρόπο οι ταξινομητές παρουσιάζουν μεγαλύτερες διαφορές μεταξύ τους και έτσι οδηγούμαστε σε καλύτερες επιδόσεις του adaboost. Έτσι συνεχίζουμε μέχρι το σ να πάρει την τιμή σ_{min}.

Αλγόριθμος AdaboostSVM

1. **Είσοδος**: σύνολο δεδομένων εκπαίδευσης με καρτέλες $\{(x_1, y_1), \ldots., (x_N y_N)\}$, το αρχικό σ: σ_{in}, το ελάχιστο σ: σ_{min}, το βήμα για το σ: σ_{step}.
2. **Αρχικοποίησή**: τα βάρη του σύνολο εκπαίδευσης: $w_i^1 = \frac{1}{N}$ για $i = 1, \ldots, N$
3. **Κάνε όσο ($\sigma > \sigma_{min}$)**
   a) Εκπαίδευσε τους ταξινομητές radial basis μηχανές διανυσματικής υποστήριξης, h_t στο ζυγισμένο σύνολο εκπαίδευσης.
   b) Υπολόγισε το σφάλμα εκπαίδευσης του h_t: $\varepsilon_t = \sum_{i=1}^{N} w_i^t , y_i \neq h_t(x_i)$.
   c) Αν $\varepsilon_t > 0.5$ μείωσε την τιμή του σ με σ_{step} και επανέλαβε το (a).
   d) Όρισε τα βάρη του ταξινομητή h_t: $a_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
   e) Ενημέρωσε τα βάρη του σύνολο εκπαίδευσης: $w_i^{t+1} = \frac{w_i^t \exp(-a_t y_i h_t(x_i))}{C_t}$
   όπου C_t είναι κανονικοποιημένη σταθερά, και $\sum_{i=1}^{N} w_i^{t+1} = 1$
4. **Έξοδος**: $f(x) = sign(\sum_{t=1}^{T} a_t h_t(x))$.

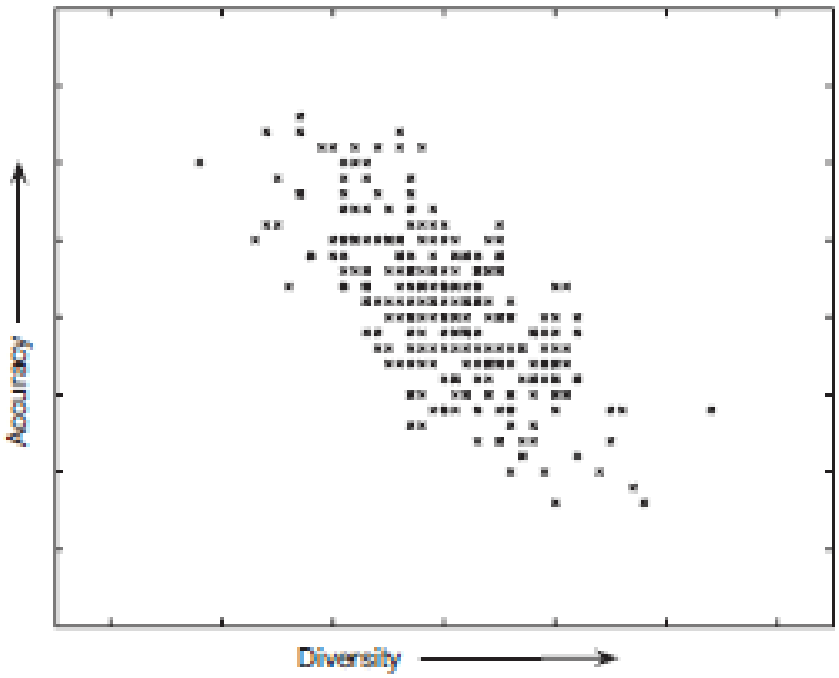
4.1.2 Ετερογενής (Diverse) adaboostSVM

Η ετερογένεια (diversity) είναι ένα ποσοτικό μέτρο που αντικατοπτρίζει το πόσα διαφορετικά είδη υπάρχουν σε ένα σύνολο δεδομένων. Η τιμή της αυξάνεται με την αύξηση των ειδών στο σύνολό μας.

Η ετερογένεια αποτελεί ένα σημαντικό παράγοντα που επηρεάζει την απόδοση του Boosting. Αυτό συμβαίνει γιατί στην περίπτωση που έχουμε ετερογένεια στους ταξινομητές μας τα σφάλματα που θα γινούν από αυτούς θα είναι ασυσχέτιστα και έτσι τα λάθη αυτά θα αφαιρεθούν με την διαδικασία της πλειοψηφικής απόφασης και έτσι πετυχαίνουμε καλύτερη γενίκευση. Στην περίπτωση του adaboost όσο πιο ακριβείς είναι στις προβλέψεις τους οι ταξινομητές τόσο λιγότερη ετερογένεια έχουμε. Αυτό γιατί αν οι ταξινομήτες μας είναι πολύ ακριβείς τότε θα διαφωνούν λιγότερα άρα τα σφάλματα θα είναι περίπου κοινά, οπότε το Boosting μπορεί να μην είναι τόσο αποτελεσματικό. Οδηγούμαστε έτσι σε ένα δίλημμα ετερογένειας – ακρίβειας. Το σχήμα 4-1 παρουσιάζει το δίλημμα αυτό. Στο σχήμα κάθε σημείο αντιστοιχεί σε ένα ταξινομητή όπου η χ - συντεταγμένη είναι η αντίστοιχη τιμή της ετερογένεια του συγκεκριμένου ταξινομητή, ενώ η y-συντεταγμένη είναι η ακρίβεια του ταξινομητή. Είναι φανερό, από το σχήμα, ότι αν ένας ταξινομητής είναι πολύ ακριβείς τότε η τιμή της ετερογένειας του είναι πολύ μικρή. Έτσι αν συνδυάσουμε ακριβείς ταξινομητές η βελτίωση που μπορεί να προσφέρει το boosting είναι πολύ περιορισμένη. Από την άλλη αν συνδυάσουμε

πολλούς ετερογενείς ταξινομητές με πολύ μικρή ακρίβεια τα αποτελέσματα μας μπορεί να είναι χειρότερα από τον συνδυασμό μερικών ετερογενών ταξινομητών και μερικών υψηλής ακρίβειας ταξινομητών. Αυτό συμβαίνει γιατί ο συνδυασμός πολλών ταξινομητών με μικρή ακρίβεια πρόβλεψης θα αποφέρει πολλές φορές λάθος αποτελέσματα ταξινόμησης και θα έχουμε κακή απόδοση.

Έτσι οδηγούμαστε στο πρόβλημα της μεγιστοποίησης της ετερογένειας υπό την προϋπόθεση να έχουμε ένα αρκετά καλό στην ακρίβεια ταξινομητή. Όπως έχουμε αναφέρει στον αλγόριθμο AdaboostSVM οι μηχανές διανυσματικής υποστήριξης με radial basis συνάρτηση πυρήνα είναι ως επί το πλείστο αρκετά ακριβείς, λόγο της κατάλληλης τροποποίησης της τιμής του σ, έτσι έχουμε την δυνατότητα να επιλέξουμε πιο ετερογενείς ταξινομητές.



ΣΧΗΜΑ 4-1 Accuracy vs Diversity

Με αυτόν τον τρόπο καταλήγουμε στον αλγόριθμο diverse Adaboost που παρουσιάζεται πιο κάτω :

Diverse AdaboostSVM

---

1. **Είσοδος**: σύνολο δεδομένων εκπαίδευσης με καρτέλες $\{(x_1, y_1), \dots., (x_N y_N)\}$, το αρχικό σ: σ_{in}, το ελάχιστο σ: σ_{min}, το βήμα για το σ: σ_{step}, ετερογένεια DIV
2. **Αρχικοποίησή**: τα βάρη του σύνολο εκπαίδευσης: $w_i^1 = \frac{1}{N}$ για $i = 1, \dots, N$
3. **Κάνε όσο ($\sigma > \sigma_{min}$)**
   a. Εκπαίδευσε τους ταξινομητές radial basis μηχανές διανυσματικής υποστήριξης, h_t στο ζυγισμένο σύνολο εκπαίδευσης.
   b. Υπολόγισε το σφάλμα εκπαίδευσης του h_t: $\varepsilon_t = \sum_{i=1}^{N} w_i^t , y_i \neq h_t(x_i)$.
   c. Υπολόγισε την ετερογένεια του h_t: $D_t = \sum_{i=1}^{N} d_t(x_i)$.
   d. Αν $\varepsilon_t > 0.5$ ή $D_t < DIV$ μείωσε την τιμή του σ με σ_{step} και επανέλαβε το (a).
   e. Όρισε τα βάρη του ταξινομητή h_t: $a_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
   f. Ενημέρωσε τα βάρη του σύνολο εκπαίδευσης: $w_i^{t+1} = \frac{w_i^t \exp(-a_t y_i h_t(x_i))}{C_t}$
   όπου C_t είναι κανονικοποιημένη σταθερά, και $\sum_{i=1}^{N} w_i^{t+1} = 1$
4. **Έξοδος**: $f(x) = sign(\sum_{t=1}^{T} a_t h_t(x))$.

Στον πιο πάνω αλγόριθμο η ετερογένεια ορίζεται ως η διαφωνία ενός ταξινομητή με όλους τους υπόλοιπους και υπολογίζεται ως εξής: αν το $h_t(x_i)$ είναι η πρόβλεψη για του t-οστού ταξινομητή για το στοιχείο x_i η $f(x_i)$ είναι η πρόβλεψη του συνδυασμού όλων των ταξινομητών, η ετερογένεια του t-οστού ταξινομητή για το στοιχείο x_i ορίζεται ως:

$$d_t(x_i) = \begin{cases} 0 \; \alpha\nu \; h_t(x_i) = f(x_i) \\ 1 \; \alpha\nu \; h_t(x_i) \neq f(x_i) \end{cases}$$

Και η συνολική ετερογένεια για τον αλγόριθμο Diverse AdaboostSVM με T ταξινομητές και N στοιχεία υπολογίζεται ως:

$$D = \frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} d_t(x_i)$$

Σε κάθε επανάληψη του αλγορίθμου η ετερογένεια D υπολογίζεται πρώτη. Αν η D είναι μεγαλύτερη από την προκαθορισμένη τιμή DIV τότε ο ταξινομητής επιλέγεται. Στην αντίθετη περίπτωση ο ταξινομητής απορρίπτεται. Με αυτό τον τρόπο δημιουργείται ένας

ακριβείς και ταυτόχρονα ετερογενής ταξινομητής. Η διαφορά του αλγορίθμου Diverse AdaboostSVM από τον απλό AdaboostSVM είναι ότι ο δεύτερος παίρνει όλους τους διαθέσιμους ταξινομητές.

4.1.3 Εφαρμογή και αποτελέσματα

Σκοπός αυτού του υποκεφαλαίου είναι να γίνει σύγκριση μεταξύ AdaboostSVM και Diverse AdaboostSVM με άλλους Adaboost αλγορίθμους που χρησιμοποιούν ως βασικούς ταξινομητές νευρώνικά δίκτυα και δέντρα αποφάσεων.

Η εφαρμογή γίνεται σε δεκατρία σύνολο δεδομένων τα οποία περιέχουν από 2 μέχρι 60 μεταβλητές. Κάθε σύνολο δεδομένων είναι χωρισμένο σε σύνολο εκπαίδευσης και σε δοκιμής με αναλογία 3/5 προς 2/5 αντίστοιχα.

Για τις μηχανές διανυσματικής υποστήριξης με radial basis συνάρτηση πυρήνα η τιμή της παραμέτρου *C* ορίζεται μεταξύ του 10 και 100 για όλα τα πειράματα. Το σ_{min} ορίζεται ως ο μέσος όρος της ελάχιστης απόστασης μεταξύ οποιονδήποτε δύο δειγμάτων του σύνολο εκπαίδευσης, το αρχικό σ_{in} ορίζεται ως η ακτίνα διασποράς του σύνολο εκπαίδευσης και τέλος το σ_{step} παίρνει τιμές μεταξύ του 1 και του 3.

Η τιμή της DIV στον Diverse AdaboostSVM ορίζεται ως ηD_{max}^{t} όπου το $\eta \, \epsilon \, (0.1]$ και όπου D_{max}^{t} είναι η μέγιστη ετερογένεια στους t προηγούμενους γύρους. Στην εφαρμογή του κεφαλαίου αυτού το η=0.7.

Στον πίνακα 4-1 παρουσιάζεται το γενικό σφάλμα με την τυπική απόκλιση των 6 αλγορίθμων στο σύνολο δεδομένων μας.

Οι 6 αλγόριθμοι μας είναι οι:

- AB_{DT} : Adaboost με δέντρα αποφάσεων για βασικό ταξινομητή.
- AB_{NN} : Adaboost με νευρωνικά δίκτυα για βασικό ταξινομητή
- AB_{SVM-s}: Adaboost με μηχανές διανυσματική υποστήριξης με παραμέτρους σταθερές για βασικό ταξινομητή
- AB_{SVM}: Adaboost με μηχανές διανυσματικής υποστήριξης για βασικό ταξινομητή
- DAB_{SVM}: Diverse Adaboost με μηχανές διανυσματικής υποστήριξης για βασικό ταξινομητή
- SVM: Μηχανές διανυσματικής υποστήριξης.

ΠΙΝΑΚΑΣ 4-1: ΓΕΝΙΚΌ ΣΦΆΛΜΑ ΚΑΙ ΤΥΠΙΚΗ ΑΠΟΚΛΙΣΗ

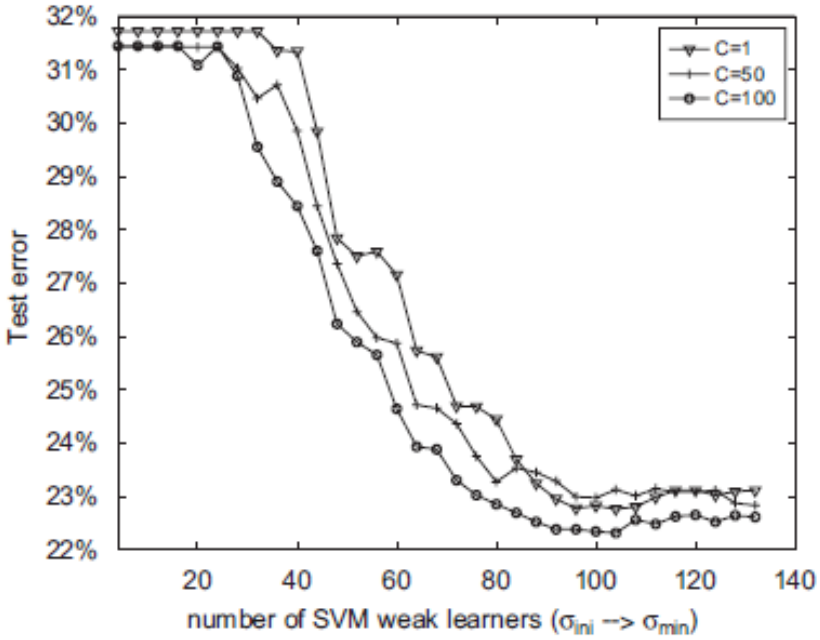
Data set	AB_{DT}	AB_{NN}	AB_{SVM-s}	AB_{SVM}:	DAB_{SVM}	SVM
Banana	13.2±0.7	12.3±0.7	14.2±0.6	12.1±1.7	**11.3±1.4**	11.5±0.7
b.cancer	32.3±4.7	30.4±4.7	30.4±4.4	25.5±5.0	**24.8±4.4**	26.0±4.7
Diabetes	27.8±2.3	26.5±2.3	24.8±2.0	24.8±2.3	24.3±2.1	**23.5±1.7**
German	29.3±2.4	27.5±2.5	25.8±1.9	23.4±2.1	**22.3±2.1**	23.6±2.1
Heart	21.5±3.3	20.3±3.4	19.2±3.5	15.5±3.4	**14.9±3.0**	16.0±3.3
Image	3.7±0.8	2.7±0.7	6.2±0.7	2.7±0.7	**2.4±0.5**	3.0±0.6
Ringnorm	2.5±0.3	1.9±0.3	5.1±0.2	2.1±1.1	2.0±0.7	**1.7±0.1**
f.solar	37.9±1.5	35.7±1.8	36.8±1.5	33.8±1.5	33.7±1.4	**32.4±1.8**
Splice	12.0±0.3	**10.1±0.5**	14.3±0.5	11.1±1.2	11.0±1.0	10.9±0.7
Thyroid	5.6±2.0	4.4±2.2	8.5±2.1	4.4±2.1	**3.7±2.1**	4.8±2.2
Titanic	23.8±0.7	22.6±1.2	25.6±1.2	22.1±1.9	**21.8±1.5**	22.4±1.0
Twonorm	3.5±0.2	3.0±0.3	5.7±0.3	2.6±0.6	**2.5±0.5**	3.0±0.2
Waveform	12.1±0.6	10.8±0.6	12.7±0.4	10.3±1.7	10.2±1.2	**9.9±0.4**
Average	17.4±1.5	16.0±1.6	17.7±1.5	14.6±1.9	**14.2±1.7**	14.5±1.5

Όπως παρατηρούμε από τον πίνακα 4-1 ο AdaboostsSVM έχει μικρότερο γενικό σφάλμα σε όλα τα σύνολο δεδομένων από τον αλγόριθμο adaboostDT (με δέντρα αποφάσεων για βασικό ταξινομητή), άρα μπορούμε να συμπεράνουμε ότι αποδίδει καλύτερα από τον AdaboostDT σε αυτά τα σύνολο δεδομένων. Το ίδιο συμβαίνει και σε σύγκριση με τον AdaboostNN, όπως παρατηρούμε το σφάλμα για τον adaboostSVM είναι μικρότερο από αυτό του adaboostNN σε 12 από τα 13 σύνολο δεδομένων. Ακόμη ο adaboostSVM αποδίδει καλύτερα και από τον AdaboostSVM-s (με σταθερή τιμή σ). Ενώ η απόδοση του είναι οριακά χαμηλότερη από αυτή των σκέτων SVM. Την καλύτερη απόδοση από όλους τους αλγορίθμους την πετυχαίνει ο diverse AdaboostSVM ξεπερνώντας τα SVM σε 10 από τα 13 σύνολο δεδομένων. Άξιο παρατήρησης είναι και το γεγονός του ότι ο adaboostSVM-s με σταθερή τιμή του σ αποδίδει χειρότερα από τα SVM. Αυτό οφείλεται στην ανάλυση που κάναμε προηγουμένως, ότι δηλαδή οι μηχανές διανυσματικής υποστήριξης με σταθερό σ αποτελούν ισχυρό ταξινομητή και έτσι είναι δύσκολο να επωφεληθούν από το boosting.

Τέλος καταλήγουμε στο συμπέρασμα ότι η ιδέα του ετερογενή Adaboost με radial basis μηχανές διανυσματικής υποστήριξης για βασικό ταξινομητή βελτιώνει την ακρίβεια πρόβλεψης και αποτελεί την πιο αποτελεσματική μέθοδο σε σύγκριση με τις υπόλοιπες αποδίδοντας αισθητά καλύτερα από τους αλγορίθμους AdaboostDT, AdaboostNN, AdaboostSVM-s και παρουσιάζοντας ελαφρώς καλύτερα αποτελέσματα από τα σκέτα SVM.

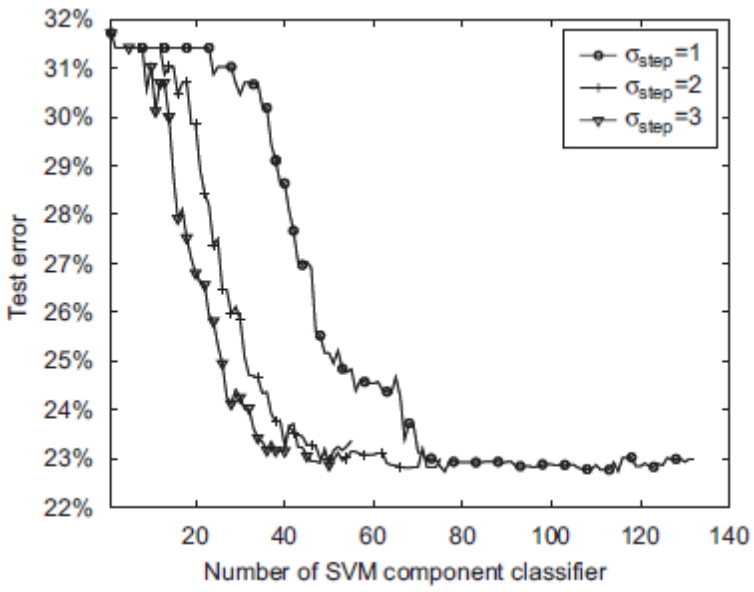
4.1.3.1 Επίδραση τον παραμέτρων C, σ_{in} ,σ_{step}

Προκειμένου να ελεγχθεί η επίδραση των παραμέτρων C, σ_{in} ,σ_{step} στον AdaboostSVM χρησιμοποιείται το σύνολο δεδομένων “titanic”. Η τιμή του C μεταβάλλεται μεταξύ 1 και 100, και εκτελούνται 100 μετρήσεις σε 100 τυχαία επιλεγμένα υποσύνολα. Στο σχήμα 4-2 παρουσιάζεται η γραφική απεικόνιση του σφάλματος ελέγχου σε συνάρτηση με τον αριθμό των ταξινομητών καθώς μεταβάλλεται το σ_{in} σε σ_{min} για *C=1*,*C=50* και *C=100*. Φαίνεται ότι για *C=100* έχουμε καλύτερα αποτελέσματα. Η διαφορά όμως είναι πολύ μικρή πράγμα που έρχεται σε συμφωνία με το ότι η παράμετρος C δεν επηρεάζει πολύ τη απόδοση των μηχανών διανυσματικής υποστήριξης με radial basis συνάρτηση πυρήνα. Για το *σ* παρατηρούμε ότι η τιμή του μειώνεται από σ_{in} σε σ_{min} όσο αυξάνεται ο αριθμός των SVM ταξινομητών. Ακόμη παρατηρούμε ότι το σφάλμα δοκιμής αρχίζει να μειώνεται μετά από το σημείο όπου το *σ* μειώνεται σε μια συγκεκριμένη τιμή, και μετά το σφάλμα αυτό αρχίζει να μειώνεται με γρήγορο ρυθμό.[30][21][20]



ΣΧΗΜΑ 4-2: Σφάλμα δοκιμής για διαφορετικές τιμές του C

Τέλος το σχήμα 4-3 απεικονίζει το σφάλμα δοκιμής σε συνάρτηση με τον αριθμό των SVM ταξινομητών για $\sigma_{step} = 1, \sigma_{step} = 2$ και $\sigma_{step} = 3$. Από την γραφική παράσταση συμπεραίνουμε ότι αν και ο αριθμός των ταξινομητών διαφοροποιείται για τις διαφορετικές τιμές του σ_{step} το τελικό σφάλμα είναι σχετικά σταθερό.



ΣΧΗΜΑ 4-3: Σφάλμα δοκιμής για διαφορετικές τιμές του σ_{step}

4.2 Bagging για μηχανές διανυσματικής υποστήριξης

Όπως έχουμε αναφέρει σε προηγούμενα κεφάλαια το bagging είναι ένας μετα-αλγόριθμος με σκοπό την βελτίωση της σταθερότητας και ακρίβειας μοντέλων ταξινόμησης. Το bagging εφαρμόζεται συνήθως έχοντας σαν βασικό ταξινομητή δέντρα αποφάσεων παρόλα αυτά στο παρών κεφάλαιο θα χρησιμοποιήσουμε ως βασικό ταξινομητή, για bagging, μηχανές διανυσματικής υποστήριξης. Οι ταξινομητές εκπαιδεύονται σε ένα Bootstrap δείγμα του σύνολο εκπαίδευσης, το οποίο παράγεται με δειγματοληψία με επανατοποθέτηση. Ο τελικός ταξινομητής κατασκευάζεται με πλειοψηφική απόφαση.

Ο χρόνος εκπαίδευσης των μηχανών διανυσματικής υποστήριξης αυξάνεται ανάλογα με τον αριθμό των δεδομένων του σύνολο εκπαίδευσης. Χρησιμοποιώντας bagging με μηχανές διανυσματικής υποστήριξης ως βασικό ταξινομητή, αποσκοπούμε στην μείωση του χρόνου εκπαίδευσης χρησιμοποιώντας λιγότερα δεδομένα εκπαίδευσης για κάθε ταξινομητή, καθώς επίσης και στην βελτίωση της ακρίβειας πρόβλεψης με την μείωση του σφάλματος.

Ο αλγόριθμός για bagged μηχανές διανυσματικής υποστήριξης καθώς και ο αλγόριθμος για δειγματοληψία παρουσιάζονται αντίστοιχα πιο κάτω:

Αλγόριθμος δειγματοληψίας

Είσοδος:

- Διάνυσμα βαρών w_i^t
- Σύνολο εκπαίδευσης $TR = \{(x_1, y_1), \dots \dots, (x_N, y_N)\}$

Διαδικασία δειγματοληψίας:

- Όρισε το δείκτη του σύνολο δεδομένων $ITR_i^t = \emptyset$
- Κανονικοποίησε το $w_i^t = \frac{w_i^t}{\sum_{i=1}^{N} w_i^t}$, υπολόγισε το διάνυσμα $C_i \;\; i = 1, \dots . N$ του συσσωρευτικού αθροίσματος w_i^t
- Κατασκεύασε ομοιόμορφη κατανομή τυχαία $R_i \;\; i = 1, \dots ., N$
- Για $i = 1, \dots, N$
  - Βρες την μέγιστη τιμή c_j (max) στο C_i η οποία είναι μικρότερη από την R_i.
  - Αν το max είναι κενό τότε $ITR_i^t = 1$ αλλίως $ITR_i^t = j + 1$

Έξοδος: $TR_t = TR_i \;\; όπου \; i = ITR_i^t$

Αλγόριθμος bagging SVM

Είσοδος:

- Σύνολο εκπαίδευσης $TR = \{(x_1, y_1), \dots \dots, (x_N, y_N)\}$ όπου x_i ϵ R^p, $y_i \epsilon\, Y = \{-1,1\}$
- SVM
- Αριθμός επαναλήψεων T
- Αριθμός των bootstrap δειγμάτων N' (N'≤ N)

Φάση εκπαίδευσης: για $t = 1, \dots, T$

- Πάρε ένα bootstrap δείγμα S_t με N' αριθμό δειγμάτων από το σύνολο εκπαίδευσης TR χρησιμοποιώντας τον αλγόριθμο δειγματοληψίας. Όρισε τα βάρη $w_i^t = \frac{1}{N}$ για κάθε επανάληψη.
- Εκπαίδευσε τις μηχανές διανυσματικής υποστήριξης στο S_t και πάρε τον ταξινομητή h_t.
- Πρόσθεσε τον h_t στο σύνολο των ταξινομητών.

Έξοδος: πλειοψηφική απόφαση για το σύνολο δοκιμής Z

$$h_f(z) = argmax \sum_{t=1}^{N} [h_t(z) = y]$$

4.3 Εφαρμογή και αποτελέσματα

Στο παρών υποκεφάλαιο θα γίνει σύγκριση των αλγόριθμων baggedSVM, adaboostSVM, arc-x4SVM, SVM καθώς και ενός τροποποιημένου αλγορίθμου του adaboostSVM του MadaboostSVM. Ο Madaboost προσθέτει στον adaboost δύο επιπλέον παραμέτρους: η πρώτη είναι η αναλογία του δείγματος f με σκοπό την μείωση του υπολογιστικού κόστους και την αύξηση της τυχαιότητας στα δείγματά μας. Και η άλλη η παράμετρος λ που εισάγεται στην διαδικασία ενημέρωσης των βαρών $(D_{t+1}(i) = D_t(i)\, \beta_t{}^{1-[h_t(x_i) \neq y_i]/\lambda})$ με σκοπό την ενημέρωση των πιθανοτήτων που αντιστοιχούν στο σύνολο εκπαίδευσης, για βελτίωση της ακρίβειας. Κατά τα άλλα ο αλγόριθμος MadaboostSVM είναι όμοιος με τον adaboostM1. Ο αλγόριθμος arc-x4 παρουσιάζεται πιο κάτω.

Αλγόριθμος Arc-x4

Είσοδος:

- Σύνολο εκπαίδευσης $TR = \{(x_1, y_1), \dots \dots, (x_N, y_N)\}$ όπου x_i ϵ R^p, $y_i \epsilon\, Y = \{-1,1\}$
- SVM
- Αριθμός επαναλήψεων T

Αρχικοποίηση: το διάνυσμα βαρών στο σύνολο εκπαίδευση TR ως: $w_i^t = \frac{1}{N}$, $i = 1,2, \dots \dots N$

Για $t = 1,2 \dots . N$:

- Κάλεσε τον αλγόριθμο δειγματοληψίας και επέλεξε ένα νέο σύνολο εκπαίδευση με δειγματοληψία με αντικατάσταση από το αρχικό TR $TR_t = \{x_i^t, y_i^t\}_{i=1}^N$
- Εκπαίδευσε τις μηχανές διανυσματικής υποστήριξης στο TR_t και πάρε τον ταξινομητή $h_t: X \to Y$
- Πάρε την κατανομή πιθανοτήτων για την επιλογή του δείγματος i που θα βρίσκεται στο επόμενο σύνολο εκπαίδευσης

$$w_i^{t+1} = \frac{1 + m_i^4}{\sum_{i=1}^N (1 + m_i^4)}$$

Έξοδος : πλειοψηφική απόφαση για το τεστ ελέγχου Z

$$h_f(z) = \arg\max \sum_{t=1}^{N} [h_t(z) = y]$$

Για την σύγκριση και αξιολόγηση τις απόδοσης των διαφορετικών αλγορίθμων, που χρησιμοποιούν μηχανές διανυσματικής υποστήριξης ως βασικούς ταξινομητής, χρησιμοποιούνται 20 σύνολο δεδομένων από το «UCI», που παρουσιάζονται στον πίνακα 4-2.

Για τις μηχανές διανυσματικής υποστήριξης χρησιμοποιήθηκαν τρεις συναρτήσεις πυρήνα: polynomial, radial basis και linear. Η τιμή του f και λ για το αλγόριθμο MAdaboostSVM ορίστηκαν ως $f = 0.8$, $\lambda = 4$. Για κάθε σύνολο δεδομένων οι αλγόριθμοί εκπαιδεύτηκαν και δοκιμάστηκαν δέκα φορές και καταγράφηκε ο μέσος όρος

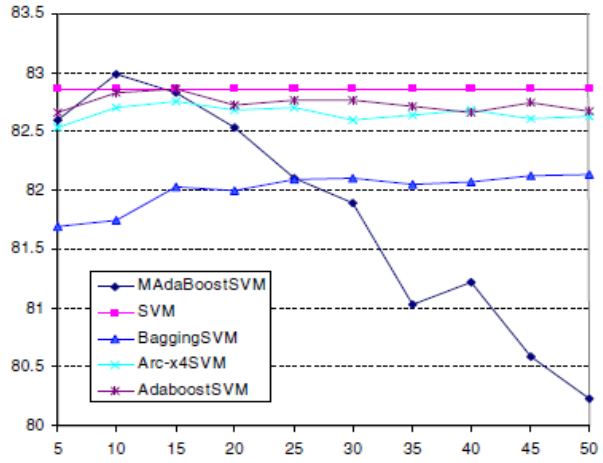
τις ακρίβειας τους. Σε όλους τους αλγορίθμους χρησιμοποιήθηκαν τα ίδια σύνολο εκπαίδευσης και δοκιμής.

ΠΙΝΑΚΑΣ 4-2: ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ

Σύνολο δεδομένων	*Στοιχεία*	*Μεταβλητές*	*Κλάσεις*
Breast cancer	683	9	2
Statlog(Australian credit approval)	690	14	2
Statlog(German credit approval)	1000	24	2
Pima Indians diabetes	768	8	2
Glass identification	214	9	6
Statlog (heart)	270	13	2
Iris	150	4	3
Statlog(Vehicle Silhouttes)	846	18	4
Connectionist bench	208	60	2
Ionosphere	351	34	2
Wine	178	13	3
Soybean	47	35	4
Vowel recognition	528	10	11
Balance scale	576	4	2
Teaching assistant evaluation	151	5	3
Image segmentation	2310	19	7
Statlog(landsat satellite)	6435	36	6
Waveform-40	5000	40	3
Letter recognition	20000	16	26
Optical recognition of handwritten digits	5620	64	10

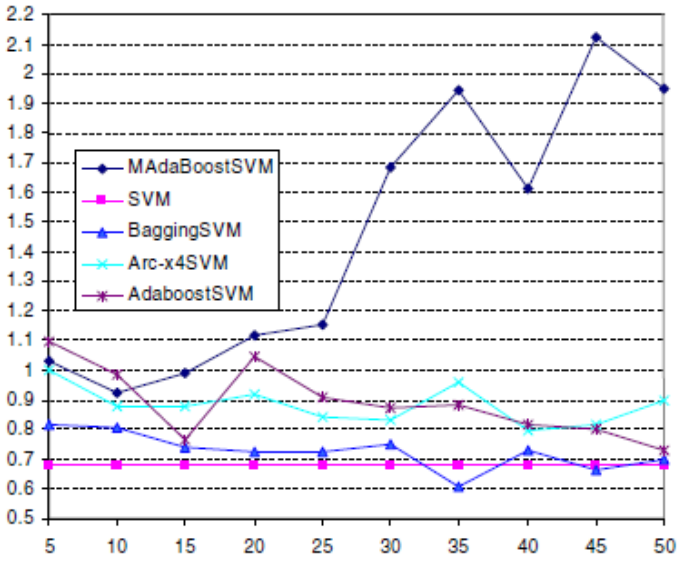
4.3.1 Αποτελέσματα-ανάλυση αποτελεσμάτων

Στο σχήμα 4-4 παρουσιάζεται η ακρίβεια των πέντε μεθόδων για 5 μέχρι 50 σύνολο SVM ταξινομητών με συνάρτηση πυρήνα την radial basis.



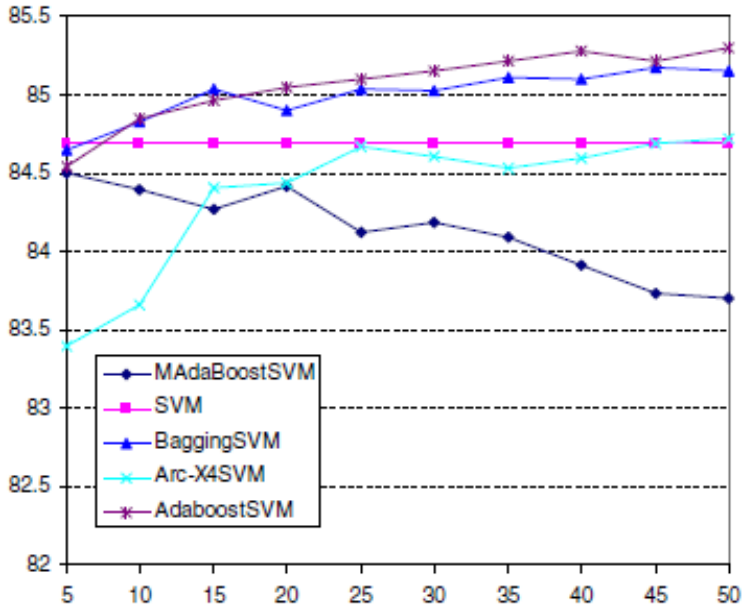
ΣΧΗΜΑ 4-4: Μέση ακρίβεια RBF SVM

Στο σχήμα 4-5 παρουσιάζεται η τυπική απόκλιση των μεθόδων μας και πάλι για για 5 μέχρι 50 σύνολο SVM ταξινομητών με συνάρτηση πυρήνα την radial basis.

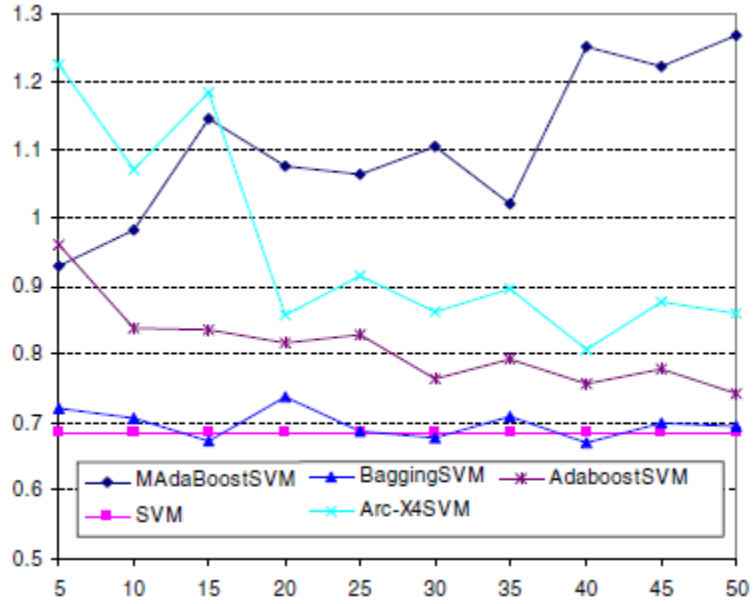


ΣΧΗΜΑ 4-5: Μέση τυπική απόκλιση RBF SVM

Στα σχήματα 4-6 και 4-7 παρουσιάζεται η ακρίβεια των πέντε μεθόδων για 5 μέχρι 50 σύνολο SVM ταξινομητών με συνάρτηση πυρήνα την linear και η τυπική απόκλιση τους αντίστοιχα.

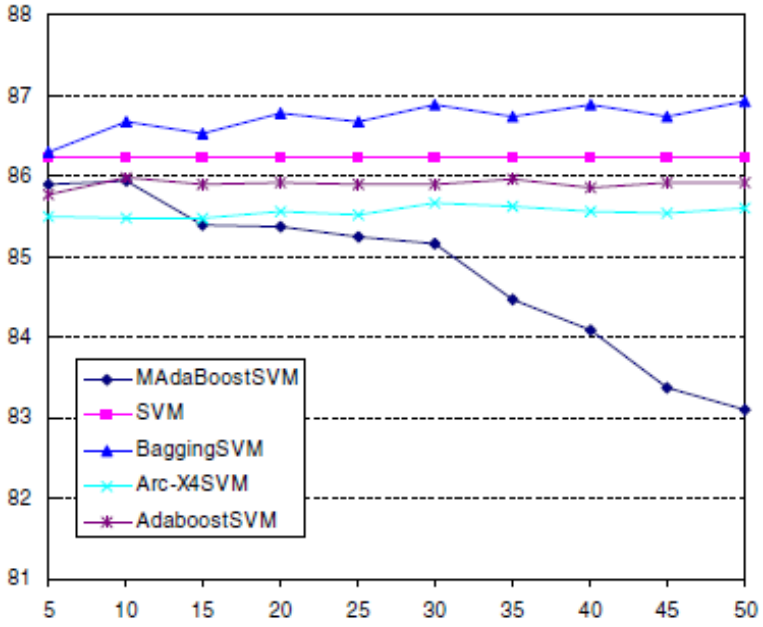


ΣΧΗΜΑ 4-6: Μέση ακρίβεια LINEAR SVM

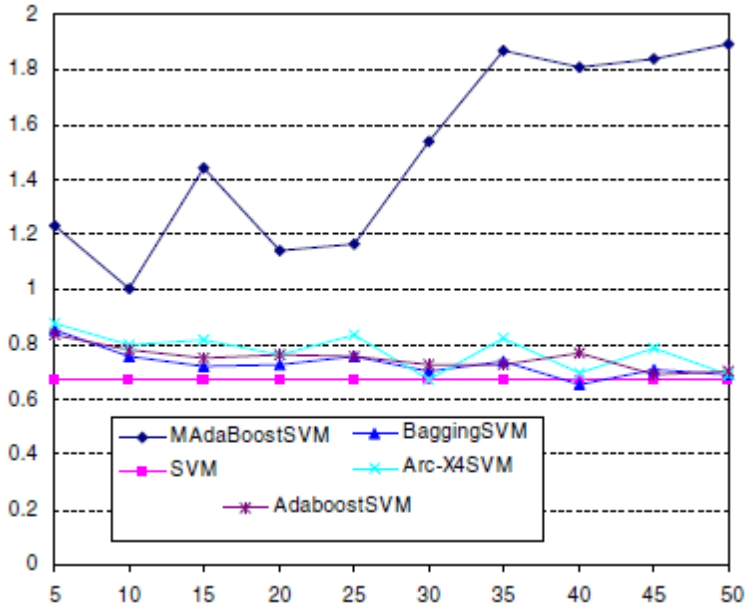


ΣΧΗΜΑ 4-7: Μέση τυπική απόκλιση LINEAR SVM

Τέλος στα σχήματα 4-8 και 4-8 παρουσιάζεται η ακρίβεια των πέντε μεθόδων για 5 μέχρι 50 σύνολο SVM ταξινομητών με συνάρτηση πυρήνα την polynomial και η τυπική απόκλιση τους αντίστοιχα.



ΣΧΗΜΑ 4-8: Μέση ακρίβεια POLYNOMIAL SVM



ΣΧΗΜΑ 4-9: Μέση τυπική απόκλιση POLYNOMIAL SVM

Είναι φανερό από τα αποτελέσματα το ότι τόσο το bagged SVM όσο και το Boosting SVM (με τις κατάλληλες τροποποιήσεις) δεν αποδίδουν πάντοτε καλύτερα από τους σκέτους SVM ταξινομητές.

Παρατηρούμε ότι στις περιπτώσεις που χρησιμοποιούμε σαν συνάρτηση πυρήνα, για τα SVM, την linear ο adaboostSVM πετυχαίνει τις υψηλότερες ακρίβειες, ενώ την μικρότερη τυπική απόκλιση την πετυχαίνουν τα σκέτα SVM. Τον καλύτερο συνδυασμό ακρίβειας τυπικής απόκλισης σε αυτή τη περίπτωση φαίνεται να την πετυχαίνουν τα baggedSVM. Ακόμη παρατηρούμε ότι ο αλγόριθμος adaboostSVM αποδίδει καλύτερα από τις δύο τροποποιήσεις του MAdaboostSVM και Arc-x4.

Στην περίπτωση όπου χρησιμοποιούμε σαν συνάρτηση πυρήνα για τα SVM την radial basis παρατηρούμε ότι τα σκέτα SVM πετυχαίνουν τις υψηλότερες επιδώσεις ενώ ο αλγόριθμος MAdaboostSVM φαίνεται να χάνει την ακρίβεια του όταν ο αριθμός των ταξινομητών αυξάνεται σε περισσότερους από δέκα.

Τέλος στην περίπτωση της Polynomial για συνάρτηση πυρήνα στα SVM το baggingSVM πετυχαίνει αισθητά καλύτερες επιδόσεις από τους υπόλοιπους αλγορίθμους.

Στους παρακάτω πίνακες 4-3, 4-4 και 4-5 παρουσιάζονται τα αριθμητικά αποτελέσματα της ακρίβειας πρόβλεψης για του αλγορίθμους μας, για τις διάφορες συναρτήσεις πυρήνα (RBF, linear,polynomial). Όπως παρατηρούμε τα baggingSVM έχουν την καλύτερη επίδοση σε 9 από τα 20 σύνολο δεδομένων στην περίπτωση του RBF, σε 8 από τα 20 στην περίπτωση του linear και σε 9 από τα 20 στην περίπτωση του polynomial.

ΠΙΝΑΚΑΣ 4-3: ΑΚΡΙΒΕΙΑ ΣΕ RBF SVM

Data set	MAdaBoostSVM	SingleSVM	BaggingSVM	Arc-x4SVM	AdaBoostSVM
Breast cancer-Wisconsin (Origin)	95.54 (0.495)	95.26 (0.415)	96.57 (0.230)	95.35 (0.440)	95.21 (0.515)
Statlog (Australian credit approval)	81.20 (0.788)	79.86 (0.588)	85.35 (0.351)	78.94 (0.969)	79.16 (0.629)
Statlog (German Credit Data)	70.85 (0.552)	70.72 (0.585)	76.41 (0.415)	70.81 (0.507)	70.85 (0.806)
Pima Indians diabetes	74.05 (0.982)	74.29 (0.816)	77.21 (0.327)	71.43 (1.151)	71.95 (0.787)
Glass identification	71.24 (1.785)	70.48 (1.250)	64.14 (2.590)	69.52 (1.770)	70.04 (1.670)
Statlog (heart)	78.30 (1.161)	79 (1.210)	83.48 (0.785)	78.56 (1.289)	78.41 (1.929)
Iris	95.47 (0.878)	95.4 (0.378)	96.2 (0.945)	94.4 (0.900)	94.53 (0.820)
Statlog (Vehicle Silhouettes)	82.77 (0.502)	82.36 (0.682)	80.02 (0.577)	82.52 (0.718)	82.54 (1.054)
Connectionist bench (Sonar)	82.25 (1.399)	81.85 (1.473)	75.85 (1.313)	81.8 (2.002)	82.55 (2.466)
Ionosphere	94.51 (0.568)	95 (0.410)	88.94 (1.035)	94.89 (0.342)	94.6 (0.392)
Wine	94.53 (1.733)	97.88 (0.568)	96.71 (0.632)	97.94 (0.747)	98.01 (0.623)
Soybean (small)	36.67 (1.889)	36.67 (1.889)	38.89 (2.147)	36.67 (1.889)	37.11 (3.482)
Vowel recognition	98.19 (1.115)	98.96 (0.226)	83.94 (0.786)	98.71 (0.273)	98.69 (0.311)
Balance scale	98.12 (0.573)	97.32 (0.446)	93.68 (0.370)	98.09 (0.374)	97.49 (0.543)
Teaching assistant evaluation	54.87 (1.913)	56.07 (2.361)	52.2 (2.201)	53.8 (2.515)	54.87 (1.751)
Image segmentation	93.86 (0.454)	93.63 (0.105)	92.28 (0.538)	93.86 (0.346)	93.61 (0.309)
Statlog (landsat satellite)	88.05 (0.457)	86.23 (0.068)	85.73 (0.270)	87.72 (0.289)	86.9 (0.289)
Waveform-40	84.95 (0.344)	83.7 (0)	86.05 (0.276)	84.47 (0.279)	84.36 (0.398)
Letter recognition	97.21 (0.077)	97.14 (0.063)	84.60 (0.177)	97.23 (0.117)	97.10 (0.168)
Optical recognition of handwritten digits	87.11 (0.809)	85.31 (0)	96.60 (0.192)	87.27 (0.683)	88.49 (0.759)

ΠΙΝΑΚΑΣ 4-4: ΑΚΡΙΒΕΙΑ ΣΕ LINEAR SVM

Data set	MAdaBoostSVM	SingleSVM	BaggingSVM	Arc-x4SVM	AdaBoostSVM
Breast cancer- Wisconsin (Origin)	96.72 (0.184)	96.69 (0.233)	**96.78** (0.224)	95.99 (0.250)	96.65 (0.217)
Statlog (Australian credit approval)	**86.07** (0.480)	85.25 (0.296)	85.23 (0.285)	83.78 (1.313)	85.68 (0.757)
Statlog (German credit data)	76.26 (0.448)	**76.59** (0.328)	76.58 (0.399)	73.36 (0.628)	76.34 (0.517)
Pima Indians diabetes	76.96 (0.418)	77.17 (0.272)	**77.24** (0.334)	72.92 (0.986)	76.86 (0.639)
Glass identification	65 (1.827)	64.90 (1.810)	**65.33** (1.502)	62.95 (2.447)	64.24 (2.075)
Statlog (heart)	83.19 (1.494)	**83.37** (0.790)	83.19 (0.841)	79.22 (1.380)	82.11 (0.891)
Iris	96.6 (0.492)	96.27 (0.783)	**96.4** (0.717)	95.2 (0.984)	95.53 (1.045)
Statlog (Vehicle Silhouettes)	80.44 (0.527)	80.27 (0.514)	80.57 (0.601)	78.99 (0.990)	**81.13** (0.774)
Connectionist bench (Sonar)	75.5 (1.732)	73.65 (2.186)	**75.55** (1.212)	75.3 (1.670)	74.85 (1.842)
Ionosphere	**88.83** (1.146)	87.6 (1.168)	88.31 (1.281)	87.17 (1.460)	87.34 (1.278)
Wine	93.06 (1.408)	96.47 (0.679)	96.82 (0.496)	**97** (0.896)	96.71 (1.007)
Soybean (small)	91.78 (4.691)	**99.56** (1.405)	99.56 (1.405)	99.56 (1.405)	99.56 (1.405)
Vowel recognition	86.27 (1.276)	82.42 (1.050)	83.96 (0.885)	**89.88** (1.162)	89.08 (0.718)
Balance scale	94.07 (0.420)	94.51 (0.438)	93.84 (0.425)	**95.37** (0.407)	95.23 (0.601)
Teaching assistant evaluation	53.73 (1.698)	**54.27** (1.265)	52.13 (1.958)	47.6 (2.884)	54 (1.176)
Image segmentation	91.40 (0.382)	**92.77** (0.173)	92.05 (0.654)	91.2 (0.851)	91.29 (0.570)
Statlog (landsat satellite)	85.21 (0.399)	85.47 (0.034)	**85.87** (0.236)	83.82 (0.530)	84.53 (0.366)
Waveform-40	85.83 (0.337)	**86.26** (0.052)	85.83 (0.298)	85.32 (0.621)	85.5 (0.583)
Letter recognition	84.32 (0.177)	83.99 (0.117)	**84.56** (0.242)	82.14 (0.446)	83.91 (0.123)
Optical recognition of handwritten digits	96.64 (0.121)	96.25 (0.105)	**96.67** (0.140)	96.49 (0.132)	96.49 (0.168)

ΠΙΝΑΚΑΣ 4-5: ΑΚΡΙΒΕΙΑ ΣΕ POLYNOMIAL SVM

Data set	MAdaBoostSVM	SingleSVM	BaggingSVM	Arc-x4SVM	AdaBoostSVM
Breast cancer-Wisconsin (Origin)	**95.31** (0.407)	94.12 (0.410)	94.85 (0.455)	94 (0.541)	94.37(0.444)
Statlog (Australian Credit Approval)	84.70 (0.767)	84.55 (0.691)	**85.09** (0.892)	81.71 (0.751)	82.54 (0.891)
Statlog (German credit data)	70.34 (0.942)	70.55 (0.937)	**71.28** (0.666)	69.81 (0.772)	68.95 (0.836)
Pima Indians diabetes	**76.42** (0.789)	75.55 (0.861)	76.18 (0.799)	72.82 (1.372)	75.83 (0.645)
Glass identification	69.62 (1.398)	69.29 (1.172)	69.57 (1.239)	70.33 (2.323)	**71.67** (1.960)
Statlog (heart)	77.70 (1.829)	75.41(1.174)	**78.48** (1.312)	76.26 (1.465)	76.89 (0.785)
Iris	96 (0.544)	**96.07** (0.734)	95.6 (0.953)	94.73 (0.378)	95 (0.786)
Statlog (Vehicle Silhouettes)	85.14 (0.400)	84.70 (0.776)	**85.15** (0.597)	83.76 (0.851)	84.93 (0.792)
Connectionist bench (Sonar)	85.15 (1.547)	**85.6** (1.792)	84.9 (2.092)	85.35 (1.248)	84.95 (0.896)
Ionosphere	87.97 (0.779)	87.46 (0.767)	**88.11** (0.635)	87.43 (0.571)	87.94 (0.795)
Wine	93.53 (1.493)	97.18 (0.992)	96.88 (0.879)	97.18 (0.823)	**97.24** (1.111)
Soybean (small)	93.11 (4.499)	100 (0)	100 (0)	100 (0)	100 (0)
Vowel recognition	96.27 (0.623)	96.15 (0.790)	95.90 (0.720)	**96.37** (0.699)	96.35 (0.505)
Balance scale	99.37 (0.3)	100 (0)	100 (0)	100 (0)	100 (0)
Teaching assistant evaluation	55.93 (1.762)	56.73 (1.647)	**56.8** (1.880)	50.67 (2.244)	55.47 (2.218)
Image segmentation	92.19 (0.401)	92.09 (0.221)	**92.52** (0.458)	91.96 (0.556)	91.46 (0.341)
Statlog (landsat satellite)	83.85 (0.949)	84.75 (0.136)	**85.94** (0.766)	81.51 (0.643)	81.43 (0.580)
Waveform-40	**84.11** (0.351)	82.24 (0.052)	83.92 (0.496)	83.08 (0.480)	82.96 (0.259)
Letter recognition	94.84 (0.146)	95.06 (0.134)	94.75 (0.132)	95.13 (0.112)	**95.18** (0.167)
Optical recognition of handwritten digits	97.35 (0.139)	97.25 (0.198)	**97.42** (0.166)	97.35 (0.182)	96.68 (1.653)

4.4 Συμπεράσματα

Στο πρώτο μέρος αυτού του κεφαλαίου έγινε σύγκριση των αλγορίθμων Adaboost και Diverse Adaboost ,με μηχανές διανυσματικής υποστήριξης(SVM) ως βασικούς ταξινομητές, με τις απλές μηχανές διανυσματικής υποστήριξης και με άλλους αλγορίθμους με βασικούς ταξινομητές δέντρα αποφάσεων και νευρωνικά δίκτια. Τα αποτελέσματα μας δείχνουν ότι το boosting χρησιμοποιώντας μηχανές διανυσματικής υποστήριξης(SVM) ως βασικούς ταξινομητές (με τις κατάλληλες τροποποιήσεις της ετερογένειας) αποδίδουν καλύτερα από ότι το boosting με βασικούς ταξινομητές δέντρα αποφάσεων και νευρωνικά δίκτυα.

Στο δεύτερο μέρος του κεφαλαίου ελέγξαμε τους αλγορίθμους bagging, Adaboost, Arc-x4 και έναν τροποποιημένο Adaboost, με μηχανές διανυσματικής υποστήριξης ως βασικούς ταξινομητές (SVM), ως προς την απόδοση τους και τους συγκρίναμε με τις απλές μηχανές διανυσματικής υποστήριξης χρησιμοποιώντας 20 σύνολα δεδομένων. Τα αποτελέσματα μας υποδεικνύουν ότι οι μέθοδοι bagging και Boosting με μηχανές διανυσματικής υποστήριξης(SVM) ως βασικούς ταξινομητές δεν αποδίδουν πάντοτε καλύτερα από τις απλές μηχανές διανυσματικής υποστήριξης (SVM). Γενικά όμως λαμβάνοντας υπόψη το μέσο όρο των αποτελεσμάτων φαίνεται ότι οι μέθοδοι bagging και Boosting με μηχανές διανυσματικής υποστήριξης(SVM) ως βασικούς ταξινομητές παρουσιάζουν υψηλότερη ακρίβεια από τις απλές μηχανές διανυσματικής υποστήριξης(SVM) με το bagging να υπερέχει έναντι των υπολοίπων έχοντας σχετικά καλύτερη απόδοση και καλύτερη προσαρμοστικότητα.

Παρόλα αυτά το ερώτημα της επιλογής μεταξύ απλών μηχανών διανυσματικής υποστήριξης(SVM) και των μεθόδων bagging και boosting με βασικό ταξινομητή μηχανές διανυσματικής υποστήριξης(SVM) δεν έχει σαφή απάντηση. Το αισιόδοξο είναι ότι με το ενδιαφέρον που αποκτούν και την μελέτη που γίνεται σε αυτό το τομέα, ο συνδυασμός bagging και boosting με μηχανές διανυσματικής υποστήριξης(SVM) με τις κατάλληλες τροποποιήσεις πετυχαίνουν υψηλότερες επιδόσεις. [1][2][11][21][30]

5. Εφαρμογή στην R πρόβλεψη της κίνησης του δείκτη FTSE 100

5.1 Εισαγωγή

Όπως έχουμε αναφέρει και στα προηγούμενα κεφάλαια το data mining βρίσκει εφαρμογές σε πάρα πολλούς τομείς. Θα επικεντρωθούμε στην εφαρμογή του data mining στα χρηματοοικονομικά και ειδικότερα στην επενδυτική στρατηγική χαρτοφυλακίου. Κρίνεται σημαντικό να επισημάνουμε ότι οι τιμές των μετόχων και των χρηματοοικονομικών συμβολαίων γενικότερα καθορίζονται με τέτοιο τρόπο ώστε να μην υπάρχει δυνατότητα κερδοσκοπίας, δηλαδή ένας επενδυτής δεν μπορεί ακολουθώντας μια επενδυτική στρατηγική να έχει απολαβές μεγαλύτερες του μηδενός ,παρόλα αυτά πάντα υπάρχει περιθώριο για κέρδος.

Στο παρών κεφάλαιο θα επιχειρήσουμε να προβλέψουμε την κίνηση του δείκτη FTSE 100. Χρησιμοποιώντας τις μεθόδους που αναπτύξαμε στα προηγούμενα κεφάλαια θα αναλύσουμε τις τιμή του δείκτη από το 1984 μέχρι σήμερα και θα προβλέψουμε κατά πόσο η τιμή του δείκτη θα πέσει η θα ανέβει. Τέλος βάση των αποτελεσμάτων μας θα κάνουμε σύγκριση των μεθόδων που χρησιμοποιήσαμε.

Ο δείκτης FTSE 100 είναι ένας δείκτης μετοχών 100 εταιριών που είναι εγγεγραμμένες στο χρηματιστήριο του Λονδίνου και που έχουν την υψηλότερη κεφαλοποίηση. Αποτελεί έναν από τους πιο σημαντικούς χρηματιστηριακούς δείκτες και χρησιμοποιείται ευρέως. Θεωρείται ως ένα ενδεικτικό μέτρο για την ευημερία των Βρετανικών επιχειρήσεων. Ο δείκτης αυτός ανήκει στον όμιλο FTSE(Financial Times Stock Exchage) που αποτελεί θυγατρική του Χρηματιστηρίου του Λονδίνου. Ο δείκτης ξεκίνησε το 1984 και διατηρείται μέχρι σήμερα.

5.2 Ανάλυση δεδομένων

Όπως έχουμε αναφέρει θα επικεντρωθούμε στην μελέτη του χρηματιστηριακού δείκτη FTSE 100. Τα δεδομένα με το ιστορικό των τιμών του δείκτη τα βρίσκουμε στο “yahoo.finance”. Η ανάλυση και επεξεργασία των δεδομένων θα γίνει με την βοήθεια της R.

Για την ανάλυση των δεδομένων θα χρησιμοποιήσουμε τα πακέτα: “xts”, “quantmod”, “adabag”, “ada”, “rpart”, “mlbench” και “e1071”. Το πακέτο “xts” το χρειαζόμαστε για να χειριστούμε τα δεδομένα μας που αποτελούν χρονοσειρά, τα πακέτα “quantmod” και “mlbench” θα τα χρειαστούμε για την ποσοτική ανάλυση των δεδομένων μας και τέλος

τα πακέτα “adabag”, “ada” και “e1071” τα χρειαζόμαστε για την εφαρμογή των μεθόδων boodting, bagging και svm. Τα εγκαθιστούμε με τις εντολές:

```
install.packages('xts')
install.packages('quantmod')
install.packages('adabag')
install.packages('ada')
install.packages('rpart')
install.packages('mlbench')
install.packages('e1071')
```

και καλούμε τις αντίστοιχες βιβλιοθήκες με τις εντολές:

```
library(xts)
library(quantmod)
library(adabag)
library(ada)
library(rpart)
library(mlbench)
library(e1071)
```

τα δεδομένα μας μπορούμε τα αποθηκεύσουμε στον υπολογιστή μας και μετά με την εντολή read() να τα εισάγουμε στην R ή μπορούμε να τα φορτώσουμε κατευθείαν από το διαδύκτιο με την εντολή:

```
ftse <- getYahooData("^FTSE", 19840103, 20140610)
```

με την παραπάνω εντολή λαμβάνουμε τα δεδομένα για τον δείκτη FTSE από το yahoo.finance για το χρονικό διάστημα από τις 3/1/1984 μέχρι 10/6/2014.

Με την εντολή

```
head(ftse)
```

```
            Open   High    Low  Close Volume
1984-01-03  997.5 1001.4  997.5  997.5      0
1984-01-04  997.5  999.5  993.3  998.6      0
1984-01-05 1007.1 1015.8 1007.1 1015.8      0
1984-01-06 1019.0 1029.3 1019.0 1029.0      0
1984-01-09 1030.6 1035.4 1030.6 1034.6      0
1984-01-10 1034.6 1034.9 1029.9 1034.3      0
```

Βλέπουμε τα 6 πρώτα δεδομένα του δείκτη FTSE που όπως παρατηρούμε βρίσκονται σε μορφή χρονοσειράς. Τα δεδομένα περιλαμβάνουν τις τιμές:

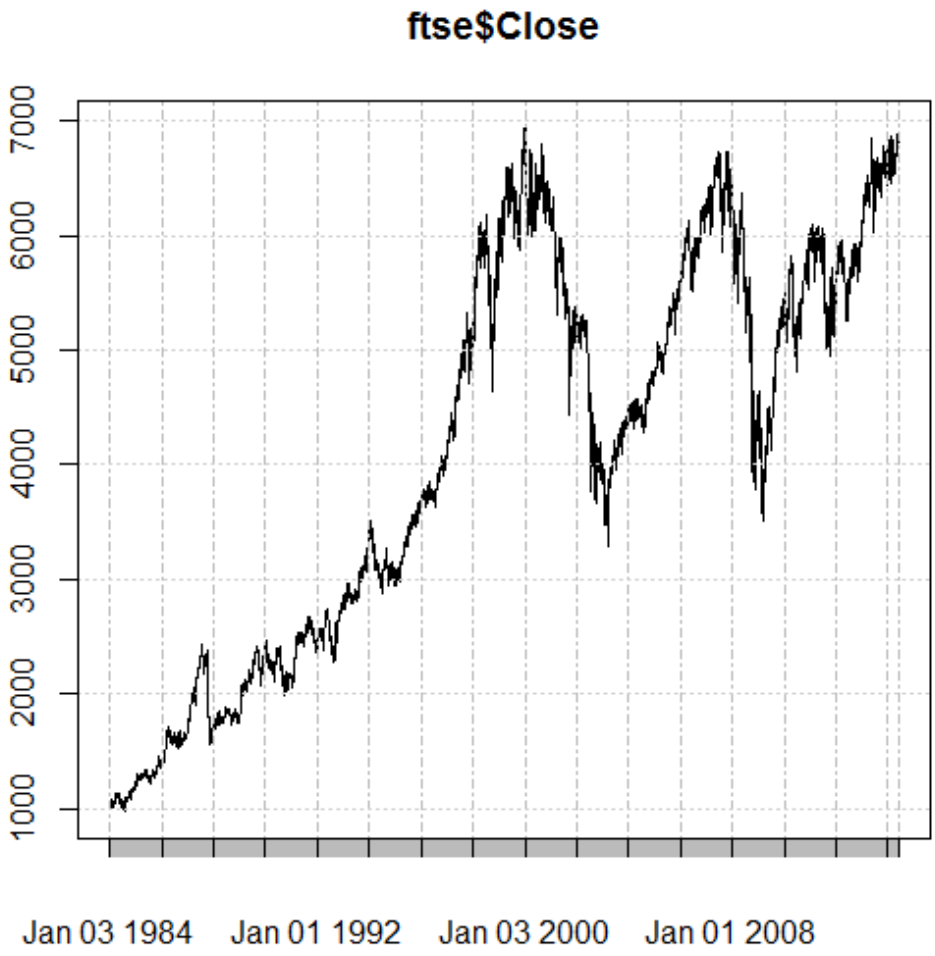
- Open που είναι η τιμή ανοίγματος του δείκτη την συγκεκριμένη ημερομηνία
- High που είναι η υψηλότερη τιμή του δείκτη την συγκεκριμένη ημερομηνία
- Low που είναι η χαμηλότερη τιμή του δείκτη την συγκεκριμένη ημερομηνία
- Close που είναι η τιμή κλεισίματος του δείκτη την συγκεκριμένη ημερομηνία
- Volume που είναι ο αριθμός των μετοχών που άλλαξαν κάτοχο την συγκεκριμένη ημερομηνία

Ακολούθως με τις εντολές:

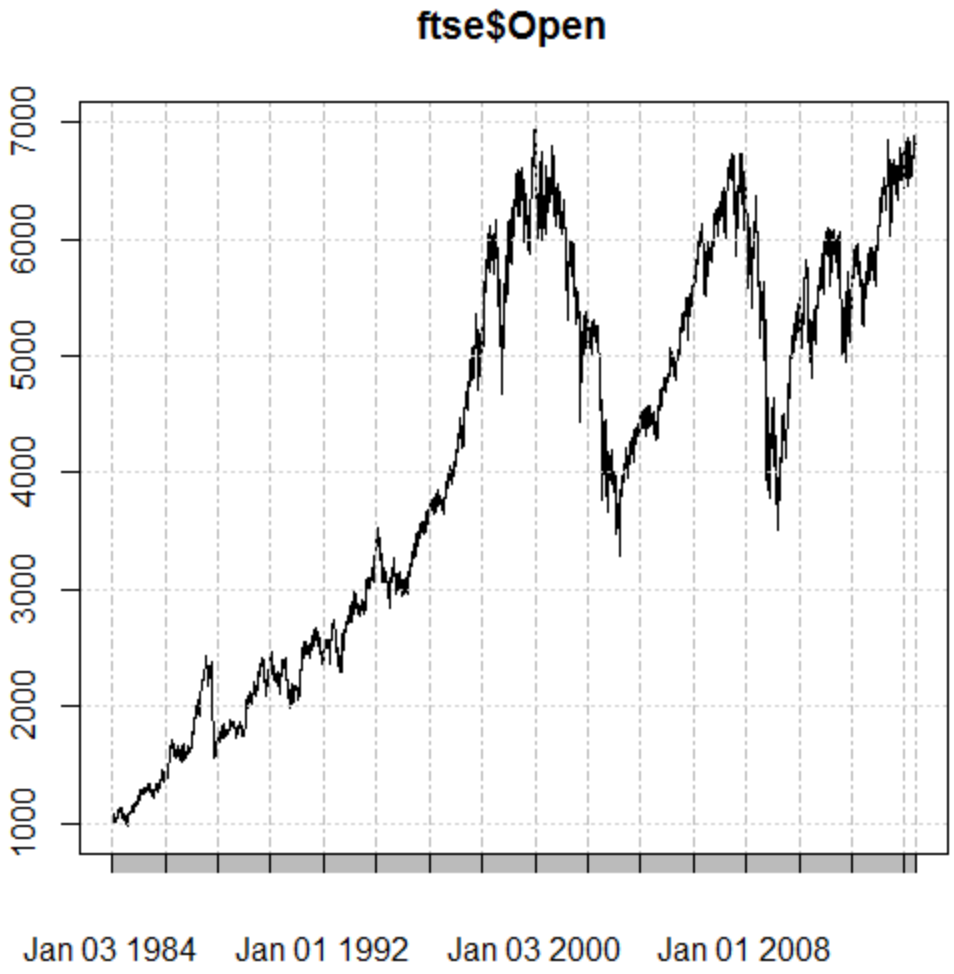
```
plot(ftse$Close, type="l")
```

```
plot(ftse$Open, type="l")
```

παίρνουμε τις γραφικές παραστάσεις των τιμών κλεισίματος και ανοίγματος του δείκτη (σχήματα 5-1, 5-2) αντιστοίχως σε συνάρτηση με τον χρόνο (σχήματα..)



ΣΧΗΜΑ 5-1: Τιμή κλεισίματος συνάρτηση του χρόνου



ΣΧΗΜΑ 5-2:Τιμή ανοίγματος συνάρτηση του χρόνου

Με την εντόλη chartSeries(ftse,subset='last 3 months') του πακέτου quantmod παίρνουμε την γραφική παράσταση του δείκτη για τους τελευταίους 3 μήνες (σχήμα 5-3). Η μπάρα με πορτοκαλί χρώμα μας δείχνει την ημερήσια μείωση ενώ η μπάρα με πράσινο χρώμα την ημερήσια αύξηση.

Με τις εντολές:

```
addCCI()
```

```
addBBands()
```

προσθέτουμε στην γραφική μας τους δείκτες “BBand” και “CCI” αντιστοίχως

- Ο “BBand” προσθέτει περιθώρια που περιλαμβάνουν τις δύο τυπικές αποκλίσεις μακριά από ένα απλό κινούμενο μέσο όρο (αναπτύχθηκε από το John Bollinger).
- Ο “CCI” (Commodity Channel Index) επιχειρεί να εντοπίσει την έναρξη και λήξη των τάσεων.

ΣΧΗΜΑ 5-3: BBAND, CCI

Όπως παρατηρούμε στο Σχήμα 5-3 η άσπρη διακεκομμένη γραμμή αναπαριστά τον μέσο όρο της τιμής του δείκτη μας. Οι πορτοκαλί και πράσινες μπάρες συμβολίζουν την ημερήσια πτώση και αύξηση της τιμής του δείκτη, ενώ οι κόκκινες διακεκομμένες γραμμές αναπαριστούν τον δείκτη BBand που περιγράψαμε πιο πριν. Κάτω από την

γραφική παρουσιάζεται το barplot του Volume. Και τέλος βλέπουμε και την γραφική του δείκτη CCI.

5.3 Μοντελοποίηση του προβλήματος

Όπως έχουμε αναφέρει στόχος της εφαρμογής μας είναι η πρόβλεψη της κίνηση του δείκτη FTSE. Ακριβέστερα θέλουμε να δημιουργήσουμε ένα μοντέλο το οποίο θα μπορεί να προβλέψει κατά πόσο η τιμή του δείκτη FTSE μια συγκεκριμένη ημέρα θα έχει άνοδο ή πτώση. Η πρόβλεψη αυτή θα βασίζεται στην εξαγωγή πληροφοριών βασιζόμενων σε προηγούμενες παρατηρήσεις (π.χ αν κάποια ακολουθία γεγονότων είχε σαν συνέπεια την αύξηση ή μείωση της τιμής του δείκτη).

Η μεταβλητή απόκρισης Y θα είναι δίτιμη (0,1) 0 για πτώση της τιμής και 1 για αύξηση. Το πρόβλημα μας είναι η επιλογή των επεξηγηματικών μεταβλητών x_i δηλαδή η εύρεση των κατάλληλων μεταβλητών που θα περιγράφουν όσο το δυνατό καλύτερα την προοπτική του δείκτη μας.

Η επιλογή των μεταβλητών αυτών θα γίνει με την βοήθεια του πακέτου της R TTR (Technical Trading Rules) . Το TTR είναι πακέτο που περιέχει συναρτήσεις και δεδομένα κατάλληλα για κατασκευή τεχνικών κανόνων συναλλαγών. Από το πακέτο TTR επιλέγουμε τις ακόλουθες για να βρίσκονται στο μοντέλο μας :

1. ATR (Average True Range): είναι ένα μέτρο μεταβλητότητας. Ο δείκτης αυτός είναι η μεγαλύτερη τιμή από τις ακόλουθες:
   - ημερήσια υψηλότερη τιμή μείον την ημερήσια χαμηλότερη τιμή.
   - απόλυτη τιμή της ημερήσιας υψηλότερης τιμής μείον την τιμή κλεισίματος της προηγούμενης μέρας.
   - απόλυτη τιμή της ημερήσιας χαμηλότερης τιμής μείον την τιμή κλεισίματος της προηγούμενης μέρας.
2. SMI: συνδέει την τιμή κλεισίματος με το μέσο της υψηλότερης και χαμηλότερης τιμής
3. ADX (Welles Wilder's Directional Movement Index): είναι ένας δείκτης που αντικατοπτρίζει την δύναμη της τάσης, δεν δείχνει κατά πόσο η τάση είναι αυξητική η μειωτική αλλά συμβολίζει την δύναμή της.
4. AROON: Ένας τεχνικός δείκτης που χρησιμοποιείται για τον προσδιορισμό των τάσεων και την πιθανότητα οι τάσεις αυτές να αντιστραφούν. Αποτελείται από δύο γραμμές: η μία γραμμή είναι η "Aroon up", η οποία μετρά τη δύναμη της ανοδικής τάσης, και η άλλη γραμμή είναι η "Aroon κάτω", η οποία μετρά την πτωτική πορεία. Ο δείκτης αναφέρει το χρόνο που παίρνει για την τιμή να φτάσει τα υψηλότερα και χαμηλότερα σημεία ως ποσοστό του συνολικού χρόνου.
5. BBANDS: είναι ένας δείκτης σύγκρισης της αστάθεια και των επιπέδων τιμών μιας μετοχής σε μια χρονική περίοδο.
6. Delt: υπολογίζει την επί τοις εκατόν μεταβολή της τιμής μιας μετοχής σε περίοδο κ ημερών

7. EMA: Υπολογίζει εκθετικά τον σταθμισμένο μέσο όρο, δίνοντας περισσότερο βάρος σε πρόσφατες παρατηρήσεις
8. MACD: συγκρίνει έναν γρήγορα μεταβαλλόμενο μέσο όρο της χρονοσείρας με έναν αργά μεταβαλλόμενο μέσο όρο της ίδιας χρονοσείρας.
9. MFI (Money Flow Ratio): Είναι ένας δείκτης ορμής που χρησιμοποιεί τη τιμή μιας μετοχής και του όγκου για να προβλέψει την αξιοπιστία της τρέχουσας τάσης. Είναι η αναλογία των θετικών ροών χρημάτων προς των αρνητικών. Για να υπολογιστεί αρχικά υπολογίζεται η τυπική τιμή ως:$\frac{high+low+close}{3}$ μετά υπολογίζεται η ροή χρήματος ως: $Typical\ price \times Volume$ ακολούθως υπολογίζεται η αναλογία ροής χρήματος (money flow ratio) ως:$(\frac{14-\text{period Positive Money Flow}}{(14-\text{period Negative Money Flow})})$ και τέλος υπολογίζεται το MFI (δείκτης ροής χρήματος) ως: $100 - [\frac{100}{(1 + \text{Money Flow Ratio})}]$. Οι θετικές ροές χρήματος δημιουργούνται όταν η ημερήσια τυπική τιμή είναι μεγαλύτερη από την προηγούμενη τυπική τιμή.
10. SAR (Stop and Reverse): εντοπίζει το σημείο stop and reverse δηλαδή το σημείο όπου μετά από αυτό η τιμή της μετοχής θα έχει πτώση.
11. Volatility: αντιπροσωπεύει την αστάθεια την διασπορά δηλαδή της τιμής.

Αφού έχουμε περιγράψει τις ανεξάρτητες μεταβλητές του προβλήματος μας τι εισάγουμε στην R με τις ακόλουθες εντολές έτσι ώστε να κατασκευάσουμε το μοντέλο μας:

```
x1<- function(x) ATR(HLC(x))[,"atr"]
x4<- function(x) SMI(HLC(x))[,"SMI"]
x3<-function(x) ADX(HLC(x))[,"ADX"]
x2<-function(x) aroon(x[,c("High","Low")])$oscillator
x6<-function(x) BBands(HLC(x))[,"pctB"]
x5<-function(x) Delt(chaikinVolatility(x[,c("High","Low")]))[,1]
x7<-function(x) EMA(CLV(HLC(x)))[,1]
x11<-function(x) MACD(Cl(x))[,2]
x10<-function(x) MFI(x[,c("High","Low","Close")], x[, "Volume"])
x8<-function(x) SAR(x[,c("High","Close")])[,1]
x9<-function(x) volatility(OHLC(x), calc="garman") [,1]
```

Όπως έχουμε αναφέρει η μεταβλητή απόκρισης μας θα πρέπει να είναι δίτιμη ειδικότερα θα παίρνει την τιμή 1 αν η τιμή του δείκτη παρουσιάσει αύξηση την συγκεκριμένη ήμερα σε σύγκριση με την προηγούμενη μέρα και 0 αν παρουσιάσει μείωση. Για να κατασκευάσουμε την εξαρτημένη μεταβλητή μας θα λάβουμε υπόψη την τιμή κλεισίματος (Close) του δείκτη μας την συγκεκριμένη ημέρα και θα την συγκρίνουμε με την τιμή κλεισίματος της προηγούμενης ημέρας. Για να το κάνουμε αυτό θα χρησιμοποιήσουμε την συνάρτηση Delt() του πακέτου “quantmod”. Όπως έχουμε πει και πιο πριν η συνάρτηση Delt υπολογίζει την επί τοις εκατόν μεταβολή της τιμής μιας μετοχής σε περίοδο κ ημερών. Ακριβέστερα υπολογίζει την τιμή:

$$\frac{C_i - C_{i-k}}{C_{i-k}}$$

Όπου C_i είναι η τιμή κλεισίματος του δείκτη FTSE την i ημέρα. Στην περίπτωση μας μας ενδιαφέρει η τιμή κλεισίματος της προηγούμενης μέρα οπότε θέτουμε $k = 1$ και η συνάρτησή μας γίνεται:

$$\frac{C_i - C_{i-1}}{C_{i-1}}$$

η συνάρτηση αυτή μας δίνει την ποσοστιαία μεταβολή της τιμής κλεισίματος της i ημέρας σε σύγκριση με την προηγούμενη, δηλαδή αν η τιμή της συνάρτηση μας είναι θετική έχουμε αύξηση της τιμής ενώ αν η τιμή της συνάρτησης είναι αρνητική τότε έχουμε μείωση της τιμής. Οπότε για την κατασκευή της δίτιμης μεταβλητής απόκρισης θα πάρουμε την τιμή της Delt για $k = 1$ και θα την μετατρέψουμε σε 1 αν είναι θετική και 0 αν είναι αρνήτικη. Αυτό γίνεται εφικτό με την βοήθεια του “ifelse” όπως φένεται παρακάτω:

```
x15<-Delt(Cl(ftse),k=1)
y <- ifelse(x15 >= '0', 1, 0)
y[is.na(y)] <- 0
```

Ανακεφαλαιώνοντας, έχουμε ορίσει τις ανεξάρτητες και την εξαρτημένη μεταβλητή μας και είμαστε έτοιμοι για να εφαρμόσουμε τις μεθόδους Bagging, Boosting και SVM που αναπτύξαμε στα προηγούμενα κεφάλαια και ακολούθως να τις αναλύσουμε και να τις συγκρίνουμε.

5.3.1 BOOSTING

Όπως αναφέρθηκε και την θεωρεία για να τη λειτουργία του boosting θα πρέπει να κατασκευάσουμε το σύνολο εκπαίδευσης και σύνολο δοκιμής. Το σύνολο εκπαίδευσης θα αποτελείται από τα $^2/_3$ του αρχικού σύνολο δεδομένων. Για την κατασκευή του χρησιμοποιούμε τις ακόλουθες εντολές:

```
l <- getYahooData("^FTSE", 19840103, 20040101)
b<-length(l[,1])
sub <- c(1:b)
```

η πρώτη εντολή καθορίζει το μέγεθος του σύνολο εκπαίδευσης που θα περιλαμβάνει τα 20 πρώτα χρόνια (από το 1984 μέχρι το 2004) και έτσι δημιουργείται το σύνολο εκπαίδευσης (sub).

Τώρα ξαναορίζουμε τις ανεξάρτητες μεταβλητές του μοντέλου μας πάνω στο σύνολο εκπαίδευσης με τις εντολές:

```
g1<-x1(ftse[sub,])
```

```
g2<-x2(ftse[sub,])
g3<-x3(ftse[sub,])
g4<-x4(ftse[sub,])
g5<-x5(ftse[sub,])
g6<-x6(ftse[sub,])
g7<-x7(ftse[sub,])
g8<-x8(ftse[sub,])
g9<-x9(ftse[sub,])
g10<-x10(ftse[sub,])
g11<-x11(ftse[sub,])
```

και ξαναορίζουμε την μεταβλητή απόκρισης πάλι πάνω στο σύνολο εκπαίδευσης:

```
x16<-Delt(Cl(ftse[sub,]),k=1)
y2 <- ifelse(x16 >= '0', 1, 0)
y2[is.na(y2)] <- 0
```

και τα μετατρέπουμε σε διανύσματα έτσι ώστε να γίνει εφικτή η λειτουργία του αλγορίθμου μας:

```
m1<-as.vector(g1)
m2<-as.vector(g2)
m3<-as.vector(g3)
m4<-as.vector(g4)
m5<-as.vector(g5)
m6<-as.vector(g6)
m7<-as.vector(g7)
m8<-as.vector(g8)
m9<-as.vector(g9)
m10<-as.vector(g10)
m11<-as.vector(g11)
y3<-as.vector(y2)
```

στην συνέχεια δημιουργούμε ένα πλαίσιο δεδομένων και εισάγουμε μέσα τις μεταβλητές μας έτσι ώστε να δώσουμε το πλαίσιο αυτό ως είσοδο στο αλγόριθμό μας:

```
data2<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y3)
y6<-as.factor(data2$y3)
data3<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y6)
```

όπως παρατηρούμε με την εντολή «`y6<-as.factor(data2$y3)`» μετατρέπουμε την μεταβλητή απόκρισης μας ως κατηγορική (0,1).

Τώρα αφού έχουμε ορίσει εξαρτημένη μεταβλητή, ανεξάρτητες μεταβλητές, σύνολο εκπαίδευσης και πλαίσιο δεδομένων είμαστε έτοιμοι να τρέξουμε τον αλγόριθμό μας. Πιο συγκεκριμένα θα χρησιμοποιήσουμε τον αλγόριθμο Adaboost από τον πακέτο “adabag” με συνάρτηση κόστους την λογιστική και θα κάνουμε 20 επαναλήψεις :

```
ftseadaboost<-
boosting(y6~.,data=data3,mfinal=20,control=rpart.control(maxdept
h=3))
```

Αφού έχουμε εκπαιδεύσει τον αλγόριθμο μας στο σύνολο εκπαίδευσης απομένει να ελέγξουμε την ακρίβεια και απόδοση του στο σύνολο δοκιμής.

Ξαναεισάγουμε τις επεξηγηματικές μεταβλητές και την μεταβλητή απόκριση όπως προηγουμένως, μόνο που τώρα έχουν ως πεδίο ορισμού το σύνολο δοκιμής:

```
v1<-x1(ftse[-sub,])
v2<-x2(ftse[-sub,])
v3<-x3(ftse[-sub,])
v4<-x4(ftse[-sub,])
v5<-x5(ftse[-sub,])
v6<-x6(ftse[-sub,])
v7<-x7(ftse[-sub,])
v8<-x8(ftse[-sub,])
v9<-x9(ftse[-sub,])
v10<-x10(ftse[-sub,])
v11<-x11(ftse[-sub,])
x17<-Delt(Cl(ftse[-sub,]),k=1)
y4 <- ifelse(x17 >= '0', 1, 0)
y4[is.na(y4)] <- 0
```

τις ξαναμετατρέπουμε σε διανύσματα:

```
m1<-as.vector(v1)
m2<-as.vector(v2)
m3<-as.vector(v3)
m4<-as.vector(v4)
m5<-as.vector(v5)
m6<-as.vector(v6)
m7<-as.vector(v7)
m8<-as.vector(v8)
m9<-as.vector(v9)
m10<-as.vector(v10)
m11<-as.vector(v11)
y7<-as.vector(y4)
```

και εισάγουμε τα διανύσματά μας σε ένα πλαίσιο δεδομένων έτσι ώστε να τις δώσουμε ως είσοδο στην εντολή predict():

```
data4<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y7)
y6<-as.factor(data4$y7)
data5<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y6)
```

Στη συνέχεια κάνουμε την πρόβλεψη μας στο σύνολο δοκιμής:

```
ftseboost.pred <- predict.boosting(ftseadaboost,newdata=data5,
newmfinal=15)
```

Με την εντολή:

```
ftseboost.pred$confusion
```

παίρνουμε τον πίνακα με τις προβλέψεις μας:

```
ftseboost.pred$confusion
                     Observed Class
Predicted Class            0    1
              0          773  366
              1          481 1099
```

Στα αποτελέσματα μπορούμε να δούμε το πιο πάνω πίνακα (Confusion Matrix) όπου παρουσιάζονται οι σωστές και λανθασμένες προβλέψεις που έχει κάνει το boosting σε 2719 δεδομένα που αποτελούν το σύνολο δοκιμής.

Όπως παρατηρούμε 773 δεδομένα ταξινομήθηκαν σωστά ως “0” (πτώση της τιμής) ενώ 366 ταξινομήθηκαν ως “0” αντί για “1” που είναι η πραγματική τιμή τους. Αντιστοίχως 481 δεδομένα ταξινομήθηκαν λανθασμένα ως “1” (άνοδος της τιμής) αντί για “0” και τέλος 1099 δεδομένα ταξινομήθηκαν σωστά ως “1”.

Με την εντολή μπορούμε να δούμε το σφάλμα πρόβλεψης:

```
ftseboost.pred$error
[1] 0.3115116
```

Το σφάλμα πρόβλεψης είναι $0.3115116 = (\frac{366+581}{2719})$.

5.3.2 Bagging

Όπως και στην περίπτωση του boosting έτσι και τώρα θα πρέπει να κατασκευάσουμε το σύνολο εκπαίδευσης και σύνολο δοκιμής. Το σύνολο εκπαίδευσης θα είναι το ίδιο με αυτό που χρησιμοποιήσαμε στο Boosting αποτελείται και πάλι από τα $^2/_3$ του αρχικού σύνολο δεδομένων. Για την κατασκευή του χρησιμοποιούμε τις ίδιες εντολές με πριν:

```
l <- getYahooData("^FTSE", 19840103, 20040101)
b<-length(l[,1])
sub <- c(1:b)
```

Τώρα ξαναορίζουμε τις ανεξάρτητες και εξαρτημένες μεταβλητές του μοντέλου μας πάνω στο σύνολο εκπαίδευση όπως προηγουμένως με τις εντολές:

```
v1<-x1(ftse[sub,])
v2<-x2(ftse[sub,])
v3<-x3(ftse[sub,])
v4<-x4(ftse[sub,])
v5<-x5(ftse[sub,])
v6<-x6(ftse[sub,])
v7<-x7(ftse[sub,])
v8<-x8(ftse[sub,])
v9<-x9(ftse[sub,])
v10<-x10(ftse[sub,])
v11<-x11(ftse[sub,])
x17<-Delt(Cl(ftse[sub,]),k=1)
y8<- ifelse(x16 >= '0', 1, 0)
y8[is.na(y8)] <- 0
```

και τα μετατρέπουμε σε διανύσματα έτσι ώστε να γίνει εφικτή η λειτουργία του αλγορίθμου μας:

```
f1<-as.vector(v1)
f2<-as.vector(v2)
f3<-as.vector(v3)
f4<-as.vector(v4)
f5<-as.vector(v5)
f6<-as.vector(v6)
f7<-as.vector(v7)
f8<-as.vector(v8)
f9<-as.vector(v9)
f10<-as.vector(v10)
f11<-as.vector(v11)
y9<-as.vector(y8)
```

δημιουργούμε το πλαίσιο δεδομένων μας και εισάγουμε μέσα τις μεταβλητές μας έτσι ώστε να δώσουμε το πλαίσιο αυτό ως είσοδο στο αλγόριθμό μας:

```
data7<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y9)
y10<-as.factor(data2$y9)
data8<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y10)
```

μετατρέπουμε την μεταβλητή απόκρισης μας ως κατηγορική (0,1).

Τώρα αφού τα ορίσαμε όλα είμαστε έτοιμοι να τρέξουμε τον αλγόριθμό μας. Πιο συγκεκριμένα θα χρησιμοποιήσουμε τον αλγόριθμο bagging από τον πακέτο “adabag” και θα κάνουμε 20 επαναλήψεις :

```
ftsebagging<-
bagging(y10~.,data=data3,mfinal=20,control=rpart.control(maxdept
h=3))
```

Με την παραπάνω εντολή εκπαιδεύσαμε το αλγόριθμό μας και τώρα απομένει να ελέγξουμε την ακρίβεια του τελικού ταξινομητή μας στο σύνολο δοκιμής.

Ξαναεισάγουμε τις επεξηγηματικές μεταβλητές και την μεταβλητή απόκριση όπως προηγουμένως, δίνοντας ως πεδίο ορισμού το σύνολο δοκιμής:

```
v1<-x1(ftse[-sub,])
v2<-x2(ftse[-sub,])
v3<-x3(ftse[-sub,])
v4<-x4(ftse[-sub,])
v5<-x5(ftse[-sub,])
v6<-x6(ftse[-sub,])
v7<-x7(ftse[-sub,])
v8<-x8(ftse[-sub,])
v9<-x9(ftse[-sub,])
v10<-x10(ftse[-sub,])
v11<-x11(ftse[-sub,])
x17<-Delt(Cl(ftse[-sub,]),k=1)
y11 <- ifelse(x17 >= '0', 1, 0)
y11[is.na(y11)] <- 0
```

τις ξαναμετατρέπουμε σε διανύσματα:

```
f1<-as.vector(v1)
f2<-as.vector(v2)
f3<-as.vector(v3)
f4<-as.vector(v4)
f5<-as.vector(v5)
f6<-as.vector(v6)
f7<-as.vector(v7)
f8<-as.vector(v8)
f9<-as.vector(v9)
f10<-as.vector(v10)
f11<-as.vector(v11)
y12<-as.vector(y11)
```

δημιουργούμε και πάλι ένα πλαίσιο δεδομένων και εισάγουμε μέσα τις μεταβλητές που ορίσαμε πιο πάνω:

```
data10<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y12)
y10<-as.factor(data10$y12)
data11<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y10)
```

Στη συνέχεια κάνουμε την πρόβλεψη μας στο σύνολο δοκιμής:

```
ftse.bagging.pred <- predict.bagging(ftsebagging,newdata=data5)
```

Με την εντολή:

```
ftse.bagging.pred$confusion
```

παίρνουμε τον πίνακα με τις προβλέψεις μας:

```
> ftse.bagging.pred$confusion
                Observed Class
Predicted Class    0    1
              0   811  451
              1   443 1014
```

όπου παρουσιάζονται οι σωστές και λανθασμένες προβλέψεις του Bagging στο σύνολο εκπαίδευσης.

Όπως παρατηρούμε 811 δεδομένα ταξινομήθηκαν σωστά ως “0” (πτώση της τιμής) ενώ 451 ταξινομήθηκαν ως “0” αντί για “1” που είναι η πραγματική τιμή τους. Αντιστοίχως 443 δεδομένα ταξινομήθηκαν λανθασμένα ως “1” (άνοδος της τιμής) αντί για “0” και τέλος 1014 δεδομένα ταξινομήθηκαν σωστά ως “1”.

Το σφάλμα πρόβλεψης τώρα είναι:

```
> ftse.baggibg.pred$error
[1] 0.3287974
```

$0.3287874 = (\frac{451+443}{2719})$.

5.3.3 SVM

Έχοντας εφαρμόσει τις τεχνικές του Bagging και Boosting έτσι ώστε να προβλέψουμε την κίνηση του δείκτη FTSE 100 στο παρών υποκεφάλαιο θα χρησιμοποιήσουμε τις μηχανές διανυσματικής υποστήριξης (SVM) για να προβλέψουμε και πάλι την κίνηση του δείκτη και ακολούθως θα συγκρίνουμε τις τρείς αυτές μεθόδους ως προς την απόδοση και ακρίβεια τους. Για να γίνει εφικτό να συγκρίνουμε τις τρεις μεθόδους αυτές θα χρησιμοποιήσουμε και στην περίπτωση των μηχανών διανυσματικής υποστήριξης το ίδιο σύνολο εκπαίδευσης και δοκιμής και τις ίδιες μεταβλητές που χρησιμοποιήσαμε στις δυο προηγούμενες περιπτώσεις , του boosting και του bagging.

Οπότε κατά τα γνωστά προχωράμε όπως και στα προηγούμενα:

Ορίζουμε και πάλι το σύνολο εκπαίδευσης ως:

```
l <- getYahooData("^FTSE", 19840103, 20040101)
b<-length(l[,1])
sub <- c(1:b)
```

συνεχίζουμε με τον ορισμό τον ανεξάρτητων μεταβλητών με πεδίο ορισμού το σύνολο εκπαίδευσης:

```
g1<-x1(ftse[sub,])
g2<-x2(ftse[sub,])
g3<-x3(ftse[sub,])
g4<-x4(ftse[sub,])
g5<-x5(ftse[sub,])
g6<-x6(ftse[sub,])
g7<-x7(ftse[sub,])
g8<-x8(ftse[sub,])
g9<-x9(ftse[sub,])
g10<-x10(ftse[sub,])
g11<-x11(ftse[sub,])
x16<-Delt(Cl(ftse[sub,]),k=1)
```

ορίζουμε και την εξαρτημένη μας μεταβλητή ως:

```
y2 <- ifelse(x16 >= '0', 1, 0)
y2[is.na(y2)] <- 0
```

μετατρέπουμε τις μεταβλητές μας (ανεξάρτητές και εξαρτημένη) σε διανύσματα:

```
m1<-as.vector(g1)
m2<-as.vector(g2)
m3<-as.vector(g3)
```

```
m4<-as.vector(g4)
m5<-as.vector(g5)
m6<-as.vector(g6)
m7<-as.vector(g7)
m8<-as.vector(g8)
m9<-as.vector(g9)
m10<-as.vector(g10)
m11<-as.vector(g11)
y3<-as.vector(y2)
```

εισάγουμε τα δεμένα μας σε πλαίσιο δεδομένων και μετατρέπουμε την μεταβλητή απόκριση από συνεχής σε κατηγορική με επίπεδα 0 και 1:

```
data6<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y3)
y6<-as.factor(data6$y3)
data7<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y6)
```

με την εντολή «svm()» του πακέτου “e1071” εκπαιδεύουμε το μοντέλο μας στο σύνολο εκπαίδευσης χρησιμοποιώντας ως συνάρτηση πυρήνα την «radial», ως μέθοδο την C ταξινόμηση ως παράμετρο gamma 0.1 (χρησιμοποιείται στην συνάρτηση πυρήνα, το 1/σ που αναφέραμε στην θεωρεία) και κόστος 10 (είναι η σταθερά κανονικοποίησης που χρησιμοποιείται στη συνάρτηση Lagrange)

```
ftse.svm<-svm(y6 ~ ., data = data7, method = "C-classification",
kernel = "radial", cost = 10, gamma = 0.1)
```

Με την εντολή «summary()» καλούμε το μοντέλο μας και παίρνουμε αρκετές πληροφορίες σχετικά με την προσαρμογή του στα δεδομένα του σύνολο εκπαίδευσης:

```
summary(ftse.svm)

Call:

svm(formula = y6 ~ ., data = data7, method = "C-classification",

    kernel = "radial", cost = 10, gamma = 0.1)


Parameters:

   SVM-Type:  C-classification

 SVM-Kernel:  radial

       cost:  10

      gamma:  0.1

Number of Support Vectors:  872
```

```
 ( 440 432 )

Number of Classes:  2

Levels:

 0 1
```

Όπως παρατηρούμε ο αριθμός των διανυσμάτων υποστήριξης είναι 872 (440 για την μία κλάση και 432 για την άλλη) ενώ έχουμε δυο κλάσεις με στάθμες 0 και 1.

Ορίζουμε ξανά τις ανεξάρτητες μεταβλητές και την μεταβλητή απόκρισης στο σύνολο δοκιμής που κατασκευάζεται από το αρχικό σύνολο αφαιρώντας το σύνολο εκπαίδευσης και αποτελείται από τα δεδομένα των τελευταίων 10 χρόνων (από το 2004 μέχρι το 2014). Και κατασκευάζουμε το πλαίσιο δεδομένων μας που περιέχει μέσα τις μεταβλητές μας:

```
v1<-x1(ftse[-sub,])
v2<-x2(ftse[-sub,])
v3<-x3(ftse[-sub,])
v4<-x4(ftse[-sub,])
v5<-x5(ftse[-sub,])
v6<-x6(ftse[-sub,])
v7<-x7(ftse[-sub,])
v8<-x8(ftse[-sub,])
v9<-x9(ftse[-sub,])
v10<-x10(ftse[-sub,])
v11<-x11(ftse[-sub,])
x17<-Delt(Cl(ftse[-sub,]),k=1)
y4 <- ifelse(x17 >= '0', 1, 0)
y4[is.na(y4)] <- 0
m1<-as.vector(v1)
m2<-as.vector(v2)
m3<-as.vector(v3)
m4<-as.vector(v4)
m5<-as.vector(v5)
m6<-as.vector(v6)
m7<-as.vector(v7)
m8<-as.vector(v8)
m9<-as.vector(v9)
m10<-as.vector(v10)
m11<-as.vector(v11)
y7<-as.vector(y4)
data7<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y7)
y6<-as.factor(data7$y7)
data8<-data.frame(m1,m2,m3,m4,m5,m6,m7,m10,m11,m8,m9,y6)
```

Με την εντολή predict() κάνουμε τη πρόβλεψη μας στο σύνολο δοκιμής:

```
ftse.svmpred <- predict(ftse.svm,data8)
```

στην συνέχεια με την εντολή table() παίρνουμε τον πίνακα ακρίβειας που μας δείχνει τα σωστά και λάθος ταξινομημένα στοιχεία της μεθόδου μας:

```
ftsesvm.acc <- table(ftse.svm.pred, y9)
```

```
                  y9
ftse.svmpred    0    1
           0   661  306
           1   574 1145
```

Παρατηρούμε ότι από τα 2686 δεδομένα 661 ταξινομήθηκαν ,με τις μηχανές διανυσματικής υποστήριξης, ορθώς ως 0, 306 ταξινομήθηκαν λανθασμένα ως 0 αντί για 1, 574 ταξινομήθηκα λανθασμένα ως 1 αντί για 0 και τέλος 1145 ταξινομήθηκαν ορθώς ως 1. Το σφάλμα πρόβλεψης γίνεται: $predicterror = \frac{306+574}{2686} = 0.327625.$

Τέλος θα ξανατρέξουμε τον αλγόριθμο μας χρησιμοποιώντας όμως σας συναρτήσεις πυρήνα για τις μηχανές διανυσματικής υποστήριξης τον “linear” και “polynomial” και θα αποφανθούμε ποιος εφαρμόζεται καλύτερα στα δεδομένα μας

```
ftse.svm.linear<-svm(y6 ~ ., data = data7, method = "C-classification", kernel = "linear", cost = 10, gamma = 0.1)
```

```
ftse.svm.polynomial<-svm(y6 ~ ., data = data7, method = "C-classification", kernel = "polynomial", cost = 10, gamma = 0.1)
```

```
ftsesvm.acc.linear <- table(ftse.svmpred.linear,y9)
```

```
ftsesvm.acc.polynomial <- table(ftse.svmpred.polynomial,y9)
```

με τις πιο πάνω εντολές παίρνουμε τους ακόλουθους πίνακες ακρίβειας:

```
ftsesvm.acc.polynomial
```

```
                          y9
ftse.svmpred.polynomial    0    1
                     0   699  377
                     1   536 1074
```

```
ftsesvm.acc.linear

                           y9
ftse.svmpred.linear     0    1
                   0  739  340
                   1  496 1111
```

ΠΙΝΑΚΑΣ 5-1: SVM

Μηχανές διανυσματικής υποστήριξης			
	Συναρτήσεις πυρήνα		
	radial	Linear	Polynomial
Predict accurancy	0.672375	0.688756	0.660089

Όπως φαίνεται στο πιο πάνω πίνακα 5-1 οι μηχανές διανυσματικής υποστήριξης με συνάρτηση πυρήνα την linear πετυχαίνουν την υψηλότερη ακρίβεια πρόβλεψης με τιμή 68.8756% ακολουθούν αυτές με συνάρτηση πυρήνα την radial και ακρίβεια 67.2375% και τέλος αυτές με συνάρτηση πυρήνα την polynomial με ακρίβεια 66.0089%.

5.4 Σύγκριση μεθόδων και εξαγωγή συμπερασμάτων

Ανακεφαλαιώνοντας έχουμε εφαρμόσει τις μεθόδους bagging, boosting και μηχανές διανυσματικής υποστήριξης (SVM) για την πρόβλεψη της κίνησης του χρηματιστηριακού δείκτη FTSE 100. Χωρίσαμε το αρχικό σύνολο δεδομένο μας σε σύνολο εκπαίδευσης, όπου εκπαιδεύσαμε κάθε αλγόριθμο ξεχωριστά, και σε σύνολο δοκιμής όπου ελέγξαμε την ακρίβεια κάθε μεθόδου χρησιμοποιώντας τις ίδιες επεξηγηματικές μεταβλητές τις οποίες πήραμε από το πακέτο «TTR».

Στο παρών υποκεφάλαιο θα επιχειρήσουμε να κάνουμε μια σύγκριση τον μεθόδων bagging, boosting και SVM που χρησιμοποιήσαμε για πρόβλεψη της κίνησης της τιμής του δείκτη FTSE 100.

Συγκεντρώνουμε τα αποτελέσματα μας στο παρακάτω πίνακα 5-2 όπου παρουσιάζεται η ακρίβεια πρόβλεψη στης κάθε μεθόδου στο ίδιο σύνολο δεδομένων:

ΠΙΝΑΚΑΣ 5-2: ΑΚΡΙΒΕΙΑ ΠΡΟΒΛΕΨΗΣ ΓΙΑ ΚΑΘΕ ΜΕΘΟΔΟ

Μέθοδος	**Ακρίβεια πρόβλεψης**
BOOSTING	0.688488
BAGGING	0.671202
SVM with radial kernel function	0.672375
SVM with linear kernel function	0.688756
SVM with polynomial kernel function	0.660089

Παρατηρούμε ότι την το bagging και οι μηχανές διανυσματικής υποστήριξης με linear συνάρτηση πυρήνα πετυχαίνουν την υψηλότερη ακρίβεια πρόβλεψης με ποσοστό 68.9% ποσοστό που θεωρείται ιδιαίτερα υψηλό αν αναλογιστούμε ότι είναι αισθητά μεγαλύτερο από την τυχαία πρόβλεψη. Οι μηχανές διανυσματικής υποστήριξης με την radial για συνάρτηση πυρήνα έχει την 3η καλύτερη επίδοση με ακρίβεια πρόβλεψης 67.23% και ακολουθεί οριακά το bagging με ακρίβεια πρόβλεψης 67.12%. Τέλος οι μηχανές διανυσματικής υποστήριξης με συνάρτηση πυρήνα την πολυωνυμική (polynomial) πετυχαίνει ακρίβεια πρόβλεψης 66%.

Συμπερασματικά τόσο το boosting όσο και το bagging όσο και οι μηχανές διανυσματικής υποστήριξης πετυχαίνουν αρκετά καλή ακρίβεια. Από αυτό συμπεραίνουμε την αποτελεσματικότητα των μεθόδων σε προβλήματα ταξινόμησης και πρόβλεψης. Ενώ κρίνεται σημαντικό να επισημάνουμε την μικρή υπεροχή του boosting έναντι του bagging στην ακρίβεια των αποτελεσμάτων.[7][9][17][15][12][26][33][13]

6. Επίλογος γενικά συμπεράσματα

Από το σύνολο των πειραμάτων που έχουμε παρουσιάσει στην παρούσα διπλωματική εργασία μπορούμε να καταλήξουμε στο συμπέρασμα ότι γενικά το Boosting θεωρείται ισχυρότερο από το Bagging πετυχαίνοντας υψηλότερη ακρίβεια πρόβλεψης παρόλα αυτά το boosting παρουσιάζει αδυναμίες σε δεδομένα με θόρυβο για τον λόγο ότι επικεντρώνεται στα λάθος ταξινομημένα στοιχεία και έτσι παρουσιάζονται φαινόμενα overfitting. Το bagging από την άλλη παρά το ότι παρουσιάζει μικρότερη ακρίβεια από το Boosting έχει σαν πλεονεκτήματα την απλότητα, την μείωση της συνδιακύμανσης, τη αποφυγή φαινομένων overfitting καθώς και το ότι μπορεί να πετύχει υψηλότερες ταχύτητες πρόβλεψης πράγμα το οποίο οφείλεται στο γεγονός του ότι το σύνολο εκπαίδευσης είναι μικρότερο λόγο της δειγματοληψίας που γίνεται. Κρίνεται σημαντικό να επισημάνουμε ότι το bagging χρησιμοποιώντας μηχανές διανυσματικής υποστήριξης ως βασικό ταξινομητή κατάφερε να πετύχει καλύτερες επιδόσεις από ότι το boosting (με τις μηχανές διανυσματικής υποστήριξης ως βασικό ταξινομητή). Αυτό μας υποδεικνύει ότι τα επίπεδα ακρίβειας που πετυχαίνει τόσο το boosting όσο και το bagging δεν είναι απόλυτα και εξαρτώνται από πολλούς παράγοντες. Οπότε το ερώτημα ποια μέθοδος είναι καλύτερη δεν μπορεί να απαντηθεί χωρίς να λάβουμε υπόψη τα χαρακτηριστικά του σύνολο δεδομένων μας. Το μόνο που μπορούμε να πούμε με σιγουριά είναι ότι τόσο το boosting όσο και το bagging όσο και οι μηχανές διανυσματικής υποστήριξης όσο και ο συνδυασμός αυτών πετυχαίνουν ιδιαίτερα ψηλά ποσοστά ακρίβειας και με την συνεχή μελέτη που γίνεται καθώς και το μεγάλο ενδιαφέρον που προσελκύουν θα παρουσιάζουν συνεχώς βελτίωση.

Βιβλιογραφία

[1]. Ahmad Ghodselahi, (2011). A Hybrid Support Vector Machine Ensemble Model for Credit Scoring. *International Journal of Computer Applications* (0975-8887), Volume 17-No.5.

[2]. Benjamin X. Wang, Nathalie Japkowicz, (2010). Boosting Support Vector Machines for Imbalanced Data Sets. *Knowledge and Information Systems,* Volume 25,Issue 1.

[3]. Chih-Wei Hsu, Chih-Jen Lin, (2002). A Comparison of Methods for Multiclass Support Vector Machines. *Transactions on Neural Networks*, VOL. 13, NO. 2,

[4]. David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang, Chih-Chen Lin, (2014). "package 'e1071' (Misc Functions of the Department of Statistics (e1071), TU Wien)".

[5]. Donglin Zeng, (2010). *STATISTICAL LEARNING AND HIGH-DIMENSIONAL DATA.*

[6]. Eric Bauer, Ron Kohavi (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine learning* 36, 105-130

[7]. Esteban Alfaro, Matias Gamez, Noelia Garcia, (2013). Adabag: An R Package for Classification with Boosting and Bagging. *Journal of Statistical Software*, Volume 54, Issue 2.

[8]. Ethem Alpaydin, (2004). *Introduction to Machine Learning.*

[9].Friedrich Leisch, Evgenia Dimitriadou, (2010). Package 'mlbench' (Machine Learning Benchmark Problems).

[10].Helge Langseth. Machine learning: Lecture 3-Bagging & Boosting + SVM

[11].Javier M. Moguerza, Alberto Munoz, (2006). Support Vector Machines. *Statistical Science*, Vol.21 No.3, 322-336 DOI:10.1214/088342306000000493.

[12]. Jeffrey A. Ryan, Joshua M. Ulrich, (2008). xts: Extensible Time Series. *R Documentation*.

[13].Jeffrey A. Ryan, (2013). package 'quantmod' (Quantitative Financial Modelling Framework). *R Documentation*.

[14].Jiawei Han, Micheline Kamber, Jian Pei, (2012). *Data Mining Concepts and Techniques,* The Morgan Kaufmann Series in Data Management Systems..

[15]. Joshua Ulrich, (2013). package TTR (Techical Trading Rules).

[16].Leo Breiman, (1996). Bagging Predictors. *Machine Learning,* 24,123-140

[17].Mark Culp, Kjell Johnson, George Michailidis. ada: an R Package for Stochastic Boosting. *Journal of Statistical Software.*

[18].Mikel Galar, Alberto Fernandez, Edurne Barrenechea,Humberto Bustince, Francisco Herrera, (2011). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-based Approaches. *Transactions on system, man, and cybernetics- part C: applications and reviews*.

[19].Peter Buhlmann, Sara van de Geer, (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications,* Springer Series in Statistics.

[20].Peter Buhlmann, Torsten Hothorn, (2007). *Boosting Algorithms: Regularization, Prediction and Model Fitting.*

[21].Ricardo Ramos Guerra, Jorg Stork, (2013*). Case Study Report: Building and analyzing SVM ensembles with Bagging and Adaboost on big data sets*.

[22].Robert E. Schapire, (2003). *Nonlinear and Classification*, Springer.

[23].Robert E.Schapire (1990). The Strength of Weak Learnability. *Machine Learning*, 5 , 197-227.

[24].Shi-jin Wang, Avin Mathew, Yan Chen, LI-feng Xi, Lin MA, Jay Lee, (2009). Empirical analysis of support vector machine ensemble classifiers. *Expert Systems with Applications 36 6466-6476*

[25].Tapas Kanungo, David M. Mount, Nathan S.Netanyahu, Christine D.Piatko, Ruth Silverman, Angela Y. Wu, (2002). An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *IEEE Transactions on pattern analysis and machine intelligence*, Nol. 24, No. 7.

[26].Terry Therneau, Beth Atkinson, Brian Ripley, (2014). package: rpart (Recursive Partitioning and Regression Trees). *R Documentation*

[27].Trevor Hastie, Robert Tibshirani, Jerome Friedman (2001). *The Elements of Statistical Learning: Data mining, Inference, and Prediction,* Springer, Second Edition.

[28].UCI Machine Learning Repository.

[29].Vapnik, V., (1997). *The nature of statistical learning theory*

[30].Xuchun Li, Lei Wang, Eric Sung, (2008). AdaBoost with SVM-based component classifiers. *ScienceDirect, Engineering Applications of Artificial Intelligence*, 21 785-795.

[31]. Yoav Freund, Robert E. Schapire., (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences.*

[32]. Yoav Freund, Robert E. Schapire, (1999). A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780

[33].Zhuo Zheng, (2006). *Boosting and Bagging of Neural Networks with Applications to Financial Time Series*.

[34].Zoubin Ghahramani, (2004). *Unsupervised Learning*.

[35].Δρόσου Π. Κρυσταλλένια, (2013). *Στατιστικές Μέθοδοι για την Ανάλυση Δεδομένων Υψηλής Διάστασης*.